Advanced Manufacturing and Automation Research Centre
Department of Mechanical Engineering
University of Bristol

Automating Skills Using A Robot Snooker Player

A thesis submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Engineering, Department of Mechanical Engineering

Shu Sang, William Cheung

April 1994

Abstract

In the modern industrial world, a wide range of industrial operations have been automated using robots. Benefits of automation include increased flexibility, higher throughput, and reduced costs. The majority of the automated operations are of a routine and repetitive nature where little or no decision making capability is required. To extend the ability of robots, it is essential that they can interact with their work environment through the assimilation of external sensory input. Furthermore, the advantages of automation would become far more widespread if human skills could be transferred into robotic systems, thereby releasing human labour from unrewarding, unpleasant and hazardous tasks.

This thesis presents the conception and development of a system that successfully integrates machine vision, human skills, and robotics. The objective is to demonstrate the feasibility of automating a complex task that demands a high level of human skill and decision making. Snooker may appear as a trivial game since human beings take for granted the ability to see, to think, and to act. Automating the playing of snooker involves a number of challenging problems.

The game of snooker is inherently random in nature i.e. there are infinite possible game situations. This can be compared to the game of chess where the number of game possibilities is finite. Therefore it is essential that the robot snooker system is able to correctly assess an unpredictable game situation. This is achieved through machine vision and an eye-in-hand visual servoing mechanism.

The human skills of feasible shot identification, judging of shot difficulty, and game planning have been analysed and incorporated into the system. A strategy has been designed and implemented to carry out a structured and organised search for the best course of action. This involves the use of a mathematical model of snooker ball dynamics so that the human player's thought process in devising a game plan can be simulated.

The system developed is capable of playing snooker fully autonomously at a skill level comparable to a competent human player. Test results show good system performance in terms of potting accuracy and the ability to play positional shots. The current system can also be used as a testbed for more advanced snooker skills and alternative game strategies.

Acknowledgements

I would like to acknowledge the help of the following : my supervisor Professor Koorosh Khodabandehloo; Terry Gorman for his electronic and mechanical input to the system development; John Byles for his technical support; Ian Rennell and Alan Jackson for their advice on snooker; and members of the Advanced Manufacturing and Automation Research Centre at the University of Bristol for their companionship.

Declaration

The accompanying dissertation entitled 'An Intelligent Robot Snooker Player' is submitted in support of an application for the degree of Doctor of Philosophy in Engineering to the University of Bristol.

The dissertation is based upon independent work by the candidate. All contributions from others have been acknowledged within the dissertation. The supervisor's contributions were those normally made in a British University.

None of the work described has been, or is being, submitted for any other degree or diploma to this or any other institute. Parts of the work have formed the basis of one conference paper.

I hereby declare that the above statements are true.

Shu Sang, William Cheung

April 1994

CONTENTS

Appendices

Figures

Tables

Chapter 1 Introduction

1.1 The Need for Automating Skilled Operations

In todays industrial world, robots are extensively utilised wherever high efficiency, productivity, and quality are demanded. Complex production processes are broken down into simple repetitive tasks which can be automated. Examples of such simple tasks include welding, riveting, paint spraying, simple pick-and-place, and loading / unloading of machine tools. In most cases, the robotic work environment is structured such that objects to be worked upon by the robot are presented in a fixed and predefined orientation. Industrial robots used in such applications generally repeat a rigid sequence of action over and over again. In Japan, where the largest population of industrial robots can be found, over 70% of the robot population are engaged in welding, material handling, and assembly tasks (World-wide Robotics Survey and Directory 1984).

As a result of advances in computer, artificial intelligence, and sensor technologies in the mid-eighties, development of robotics also move into a more advanced age. Recent statistics on the world-wide robot population show a significant shift in robot utilisation. Statistics on new robot installation in many industrial countries show that an increasing proportion of robots are employed in complex tasks that were previously beyond the capability of robots. They are commonly known as ' *intelligent robots* ' because of their ability to interact with their work environment and to function with a high degree of autonomy.

1.2 Example Skilled Robotic Systems

Unusual developments include robot Ping-Pong player **[Andersson89]**, sheep-shearing robot **[Trevelyan89]**, robots that can milk a cow **[Frost90]**, robots that can climb buildings **[Luk91]**, and mobile robots that can navigate autonomously **[Crowley85]**. Many of these novel robotic applications lead to very useful spin-offs. For example, a wall climbing robot could be modified to carry out inspection and maintenance of buildings and pipe crawling robots can check for leaks. In almost all of these alternative robotics applications, the capabilities of robots are expanded. In most intelligent robotic applications, the ability and skill of a robot are enhanced through the incorporation of external sensory inputs such as tactile or visual sensing. Information collected through sensors facilitates the interaction between robot and its work environment, which is often highly variable. As these sensory inputs are received by the robotic system, adaptations to the changes in the work environment can be made thus facilitating the successful completion of a task.

A good example of the benefits of expanding the spread of automation is the food-related industries which include meat cutting, deboning, and packaging **[Purnell90, Khodabandehloo90, Kassler90]**. The population of labour employed in the European food sector is about 2,500,000 producing over 240,000,000,000 ECU's worth of consumed food **[Khodabandehloo91]**. Automation of this industry will have a significant social and economic impact, particularly in countries such as Denmark whose major export is meat. The need for automation in the meat industry is compounded by the shortage of skilled labour and increasing demands on the quality and hygiene of meat products.

1.3 Definition of Skilled Robots

A definite trend in robot development towards skilled operations is emerging **[Birk81, Nitzan85]**. Skill is the ability to perform a specific task expertly and well. A definition of skilled robotics has been presented by Khodabandehloo as :

> **A robot can be said to have skill rather than intelligence if it demonstrates the necessary sensory perception and dexterity by performing a specific task, the output of which appears indistinguishable from that which can be produced by a human. [Khodabandehloo91]**

The ultimate aim of any skilled robotic system is to reach the same level of performance as that of a human expert in a specific field. Tasks that still need to be carried out by human workers invariably require specialised skills. Although many industrial operations are already automated, many others demand considerable skills which are difficult to quantify and usually not directly transferable to robotic systems. Research in the field of skilled robots aims to establish the necessary understanding for future developments of automation so that the benefits of automation can be more widely spread.

1.4 Demonstration of the Skill Potential of Industrial Robots through Snooker

Human tasks or games such as snooker provide a suitable test bed for research purposes. The game of snooker is played on a rectangular table with pockets at each corner and the middle of the long edges, i.e. 6 pockets in total. A full size snooker table measures 12 ft. by 6 ft., but an approximately quarter-size table measuring 5 ft. 7 in. by 2 ft. 11 in. is used in this research.

At the simplest level, the aim of the game is to hit the cue ball with a cue such that a target ball, on being hit by the cue ball, drops into a pocket. The first complication is that there are two types of target ball : red balls and colour balls. On a full size snooker table, 15 red balls are played. On the quarter-sized table used in this research, only 10 red balls are played. Each red ball is worth 1 point if potted. There are 6 colour balls to play, the points they carry are in brackets : yellow (2), green (3), brown (4), blue (5), pink (6), and black (7). The rule is to alternately pot any red ball followed by any colour ball until all red balls are potted. Note that a potted red ball is removed from play while a potted colour ball is put back on to the snooker table. When all red balls are potted, the colour ball carrying the least points is to be potted and removed from play, followed by the next etc. until all colour balls are potted. Chapter 6.6 contains a detail account of all the possible potting sequence. Each time a player pots a ball, he scores the number of points carried by that ball. Needless to say, the player with the highest score at the end of the game is the winner. Thus the highest points scoring potting sequence is to pot the black ball after a red, which is what all professional snooker players try to achieve. It has to be stated that the above is only a superficial account of the game. Nevertheless, any reader unfamiliar with the game should now have a basic understanding of snooker. Of course there are many other complications in snooker rules such as touching ball, fouls, and penalties of potting the wrong ball, etc. but unfortunately it is beyond the scope of this thesis to give a full account of the rules. Inquisitive readers can refer to publications such as **[Davis 88]**.

Research work towards the development of an Intelligent Robot Snooker Player has been carried out at the University of Bristol since 1987. It was conceived as a means of demonstrating the feasibility of robot application in complex tasks requiring skill. A pilot system capable of playing snooker autonomously was in operation by 1989 **[Khodabandehloo87]**. However, the pilot system could only play in a shot-by-shot manner i.e. no game strategy was devised. As a logical step forward, the system has been further developed to incorporate the ability to assess a game situation and plan its course of action, very much like what a professional snooker player does. This will be achieved through the implementation of a one-step ahead planning strategy and accurate ball control by the robot. Once a solid foundation is formed, the system could be modified to look further ahead or adopt alternative game strategies.

A standard industrial robot, a linear drive system, a machine vision system, 2 CCD cameras, and software encompassing expertise in snooker have been integrated into an intelligent robot snooker player system. The target is to have a robot snooker player that performs at a level comparable to a professional snooker player. It is hoped that success of the robot snooker player would demonstrate that other human skills could be similarly analysed and automated.

Automating snooker may initially appear to be divorced from the process of automating industrial tasks, but this is far from reality. Playing snooker is a skilled task for many reasons. The play area is well contained and yet there are infinite game situations. The identification and execution of a shot involves planning, reasoning and strategic decision making. The outcome of potting a ball is also highly uncertain with regard to the resulting new game situation. The system also demands a high degree of manipulator dexterity for the purpose of accurate cue positioning.

The basic skills in snooker can be broken down in the following levels :

(i) correct identification of ball colours and position

(ii) determination of whether a ball can be potted

(iii) given a feasible shot, estimate the correct shot angle

(iv) ability to judge the relative difficulties of possible shots

(v) ability to devise a game strategy to maximise the score

Some of the above skills may seem trivial for human beings. For example, skill (i) above is simply not a problem for anyone with normal vision. But for a machine, particularly one equipped with black and white vision, the task is indeed very challenging.

Performance in snooker is easily assessed : skill is proportional to the magnitude of breaks, the total score in a frame, and of course the number of frames won. Planning is crucial to success in the game since players take turns to play. So if the opponent is left with a good situation, it could be difficult to maintain or regain the lead. The game also presents a number of challenging problems to a robot system which must make decisions in a contained environment of disorder and unpredictability in order to perform well.

On the surface, snooker appears to be a 2-dimensional game. However, that is only true with respect to the playing area and ball positions. Professional snooker players manage to control the cue ball by hitting it at the desirable point often inducing top-spin, back-spin, side, or a combination of the three effects. There is also infinite variations of cueing force delivered to the cue ball.

Amateur snooker players tend to concentrate on the potting of a single ball. A professional is more concerned with the positioning of the cue ball after potting a ball. So if the skill level of the robot snooker player is to be raised, it has to be able to control the cue ball. Ball control is a most vital prerequisite for successful strategic play. Only when ball control is achieved can one anticipate the consequences of various possible pots, plan upon the anticipated game situation, and choose a shot which is likely to lead to another pot.

Professional players are known to think at least 3-steps ahead when playing a shot. Before a shot is taken, they build up in their mind the game scenario after potting a certain ball using a particular cue force. Using their past experience in playing snooker, an accurate prediction of the resulting ball positions can be made. In other words, by choosing which ball to pot into which pocket at a certain cue force, the player can control the whereabouts of the cue ball. Obviously the player is going to play the first shot in such a way that the cue ball will end up in a position where another ball can be potted. This is 1-step ahead game planning. 3-step ahead game planning is achieved by going through this predict-and-plan process 3-ply deep. Theoretically the greater the number of forward steps determined, the higher the system performance becomes. This will, however, depend on the precision of the robot and the cueing action.

1.5 System Components

The robot snooker playing system hardware is a collection of many components and sub-systems. Each component performs a specific task and by harnessing the functioning of the system components with communication and control software, the ultimate task of intelligent and strategic play of snooker is achieved.

System components such as the PUMA robot arm, the SKF Linear Driver and the Automatix Vision System are complex pieces of equipment in their own right and each of them have their own associated installation and operation manuals which are voluminous. There is no intention of reproducing these manuals in a concise form here. However an attempt is made to point out their basic operations and how they fit into the robot snooker player system.

1.5.1 Lighting Set-up

A uniform level of illumination across the snooker table is vital to the functioning of both the overhead and on-board camera in the system. The fact that the image processor works on gray-scale images of 64 gray-scale levels further emphasises the importance of this requirement.

In a previous lighting set-up, four 1000 watts quartz halogen theatre lamps had been hung in line along the length of a snooker table above it to provide illumination. The level of illumination provided is adequate but the uniformity is not good and there are some undesirable side-effects. Up to 70% of the power consumed by the halogen lamps is dissipated as heat which can be damaging to other system components such as the CCD cameras. The construction of the theatre lamp housing allows this heat to radiate in all directions. Indeed the heat produced is sufficient to raise the temperature of the object being illuminated significantly. Other disadvantages associated with this type of lighting is their relatively limited life (rated average life of 750 hours), low efficiency, heavy weight and bulky size.

After careful consideration, it is decided that tungsten halogen dichroic reflectors are most suitable. The principal of light generation is the same as that of the theatre lamps but this is as far as their similarity goes. In simple terms, light produced by the tungsten halogen interaction contains both visible and infrared lights. The function of the dichroic reflector is to allow the infrared component together with the associated heat to pass through while reflecting the cool visible light onto the scene. Another advantage of tungsten halogen dichroic reflectors is their lumen maintenance of almost 100%, i.e. there is virtually no loss in brightness as the reflector ages. The light has good colour rendering properties, is energy efficient and has a rated average life of 3000 hours. Tungsten halogen dichroic reflectors belong to the class of low voltage lighting, working at 12 V d.c. with a power rating of 50 watts each. An individual reflector is very compact, the reflector cup having a diameter of 50.7 mm, and construction is robust.

Having justified the choice of light source, the number of reflectors to be used and their arrangement have to be determined. The overhead lighting arrangement which illuminates the entire snooker table will be dealt with first. In order to obtain an image with minimal distortion, the overhead camera is to be mounted as far away from the table surface as possible. To avoid the camera view being obscured by the reflectors or the camera casting a shadow onto the scene, the reflectors are to be mounted level with the lens of the camera. Because of existing fluorescent lighting in the laboratory, the overhead camera and dichroic reflectors are to be fitted at 1.86m above the snooker table surface.

The objective of the overhead lighting grid is to produce an intense and uniform level of illumination so that minor variation of ambient lighting will have virtually no effect on scene illumination. A reflector on its own at 1.86 m from the table will produce a luminance contour the shape of a cone, as shown in figure 1.1. If a second reflector is introduced to supplement the first, the shape of the luminance contour will depend on the separation between the two reflectors. The contour will show two peaks if they are well separated or a single peak if they are close to each other. The aim is to find the separation such that the difference in luminance intensity of the two peaks and the trough in between is minimal i.e. the two peaks and the trough should merge together forming a plateau.

To find the ideal separation, two reflectors mounted 600 mm apart at 1.86 m from the table are switched on and the luminance contour is noted. At this separation, two distinct peaks show up in the contour. By gradually reducing the separation in steps of 25 mm and noting the resulting contour, the ideal separation is found to be 250 mm. Note that this separation is only correct when using dichroic reflectors of 50 watts rating and a beam angle of 21 degrees. By induction, a uniform linear illumination can be produced if a series of reflectors are line up at 250 mm intervals. Treating this linear series of reflectors again as a single unit of light source, the lighting system can be extended into 2 dimensions by arranging several of these linear series of reflectors also at 250 mm intervals. The resulting lighting grid is thus a matrix of reflectors where each reflector is 250 mm away from its four immediate neighbours.

The dimension of the snooker table is 890 mm by 1750 mm from edge to edge. Therefore a 5 by 8 matrix of tungsten halogen dichroic reflectors will cover the table surface adequately. The overhead camera is mounted at the centre of the lighting matrix. The combined wattage of this lighting system is 2000 watts. Because of their low voltage requirement, four transformers (each supplying power to 2 columns of 5 reflectors) are installed. Also, each transformer is connected to the mains supply via a dimmer switch to facilitate adjustments to be made in order to ensure a uniform level of illumination across the snooker table. Figure 1.2 is a photograph showing the overhead lighting system illuminating the snooker table.

Lighting for the on-board camera mounted to the robot wrist is taken care of by 2 miniature tungsten halogen dichroic reflectors. They are scaled down versions of the reflectors used in the overhead lighting system and have a reflector diameter of 35.3 mm and a lower output of 20 watts. A much reduced amount of lighting is sufficient for the on-board camera because the viewing distance is only 297 mm compared to 1.87 m for the overhead camera. Also a very small viewing area needs to be covered by the on-board camera and an exceedingly intense level of illumination will saturate the CCD of the camera.

1.5.2 PUMA 560 Robot Arm

The Unimate PUMA robot is a computer-controlled robot arm manufactured by Unimation (Europe) Ltd. The robot arm used in the robot snooker player system is a PUMA series 560 which has 6 degree of freedom and all joints are of the revolute type. The basic robot system is composed of the robot arm, a control cabinet which contains the robot controller, software, the CRT display and keyboard, a floppy disc drive and a manual control hand pendant.

Each member of the arm assembly is driven by a permanent-magnet dc servo motor driving through its associated gear train. Each motor in the arm contains an incremental encoder and a potentiometer. Control of the position and velocity of each joint is crucial to the proper functioning of the robot arm, and each joint is monitored by its dedicated microprocessor. Position of the robot arm at any time is derived from the 6 joint encoders in the robot arm. Encoder readings are transmitted from the servo motors to the controller to provide closed-loop control of the arm motions.

Robot positions can be stored as transformations in a co-ordinate system fixed relative to the stationary robot base or as precision points where joint angles are recorded. The robot arm can be manually controlled using the hand pendant, or by program control. The system programming language is called VAL-II which is an interpretative language. The language structure is similar to FORTRAN with additional robot control commands and transformation manipulation functions. In the robot snooker player system, the PUMA robot arm is under program control at all times and no human control is required. VAL-II programs are created using the system editor and can be stored onto any VAL-II formatted 5 $\frac{1}{4}$ inch floppy disc.

Interfacing channels are provided as 32 input (WX) and 32 output (OX) digital lines and external systems can interact with the robot via these lines. In the robot snooker player system, the PUMA controller commands the SKF linear driver and the pneumatic cue strength and firing mechanism. No dedicated communication port is present but the terminal port, which is normally connected to the CRT and keyboard, can be connected to an external device which then takes over control of the PUMA.

1.5.3 SKF Linear Driver

The SKF Linear Drive System is incorporated into the system to supplement the operation of the PUMA 560 robot arm. Its function is to station the PUMA robot arm at various fixed points to extend the PUMA 560 robot arm's effective work envelope. This is necessary because the work envelope of a stand alone PUMA 560 is not large enough to cover the snooker table. The linear drive system has 2 linear axes (X and Y) of motion with an overhead platform onto which the robot arm is mounted upside-down, see figure 1.3.

Stepping motors achieve high accuracy and repeatability through the stepwise rotation of the motor shaft. A full motor shaft revolution is equivalent to a fixed number of individual steps. The number of steps that forms one revolution depends on the design of the motor and reflects the accuracy of the stepping motor. By applying control pulses to a stepping motor, positional accuracy of 1 step without feedback is achieved. The lack of feedback is not as great a handicap as may be imagined. Without feedback, the system functions as a pure feed-forward control chain. This keeps the system simple and effective without having to deal with any servo-related problems such as stability.

The two stepping motors in the SKF are controlled by a Berger Lahr Posab 2001 positioning and sequence controller. The controller supports up to 3 stepping motors, with 16 input and 16 output digital signal lines, 255 flags and 999 variables. In this particular system, no Z-axis stepping motor is installed. The ranges for the X-axis and Y-axis stepping motors are 75838 and 36011 steps respectively. Using an LVDT to measure the movement of the X-axis and Y-axis motors, the ranges are equivalent to approximately 2.39 m in the X direction and 1.14 m in the Y direction.

With the PUMA robot arm mounted to the SKF platform, positioning of it can be under manual or program control. The program language is rather crude syntactically. For example, all assignment and arithmetic operation are either single-operand or double-operand, which is rather similar to programming using an assembly language. Nevertheless, the sole purpose of the SKF linear driver is to position the PUMA at 1 of 8 possible pre-defined points and a simple program is written to perform this task. Communication with other equipment is through the 16 input and 16 output lines. In the robot snooker player system, the SKF is under direct control by the PUMA controller.

1.5.4 Automatix Vision System

In the robot snooker player system, all digital image analysis and processing tasks are performed by the Automatix AV-5 Vision System. It is a gray-scale vision system capable of storing a maximum of fourteen 256×512 pixel 64 gray-scale image buffers. The number of available buffers can be increased by installing additional vision boards.

The system contains of 5 major components :

(1) CP 464 Vision Processor

The CP 464 runs two 12 MHz 68000 microprocessors. Two mega-byte of on-board memory is available and a maximum of 14 pixel buffers can be stored. In case not all 14 buffers are required, the corresponding memory can be released for use by the CP 4932 control processor.

The AV-5 can support an extra CP 464 vision processor in which case one acts as the master and the other the slave. With such an arrangement, an extra 14 pixel buffers will be available. No slave CP 464 is currently present in the vision system.

(2) CP 4932 Control Processor

This is the processor that controls program execution and non-vision related computations. One mega-byte of on-board memory exists but any unused memory from the CP 464 can be utilised by the CP 4932. Again, a 68000 microprocessor is the heart of the control processor. Both the keyboard and CRT display unit communicates directly with the CP 4932 control processor.

(3) Disk File Storage System

The AV-5 incorporates two 3 ½ inch double-sided floppy disk drives. Since the optional 20 MB Winchester hard disk drive is not fitted, booting up has to be done via the floppy disk drives.

(4) System Software

A copy of system program called AIMON is resident in every 68000 microprocessor in the system. It is a program that is executed every time the vision system is started up. Its main functions are to perform power-up test, system initialisation, and to load and execute the RAIL operating system. AIMON is a background environment that the user need not concern himself with. The user normally works in the RAIL operating system at all times unless a system failure occurs, in which case control is returned to AIMON where system diagnosis can be made.

RAIL is the operating environment in which user written programs execute and software is written in RAIL version 6.04 . This version of RAIL is more versatile than earlier versions in many ways. Apart from some additional program structures e.g. the CASE statement, CCD cameras of different makes can be configured into the vision system. This is not possible with previous versions of RAIL which only support one type of camera e.g. the Matsushita BS 703 CCD camera.

(5) CCD Cameras

Two CCD cameras are configured into the vision system. A Matsushita BS 703 is used to provide the vision system with an overhead view of the entire snooker table. A second camera, the Pulnix TM540 with remote head, is fitted to the wrist of the PUMA robot arm as part of the visual servoing system. Each camera has only a single pixel buffered allocated so that the maximum amount of memory can be released for use by the CP 4932 control processor.

The AV-5 vision system has a single external RS-232 port for communication with other equipment. As will be seen in the next chapter, control of the PUMA by the AV-5 is conducted through this port.

1.5.5 Pneumatic Snooker Cue and Control

In the robot snooker player system, the functioning of the vision system, and shot selection and planning modules culminates in the actual physical potting of snooker balls. It is assumed here that the reader understands the basic cueing action, that of forming a bridge with the left hand and stroking the cue with the right hand for a right-handed player. The strength of a shot is controlled by varying the length of the stroke.

It is possible to emulate this sequence of actions using the PUMA robot arm. A short piece of snooker cue can be fitted as the robot end-effector. Thus equipped, the robot can take a shot by performing a straight line move at the cue ball in the required direction. However, there are three serious problems with such a robotic cueing action. Problem number one is that of robot kinematics. Basically, the system has to have the capability of hitting the cue ball at any point on the snooker table in any direction. The two points defining the stroke of the cue can be easily computed in robot co-ordinates. However, the robot is not always possible to make such a simple straight line move because of joint limits being reached, or a necessary change of robot posture. Problem number two is the speed at which the straight line move can be made, if such a move is kinematically possible. To play a powerful shot, the robot will have to perform a rapid linear motion. But it is a well known fact that joint singularities will impose a limit to the resultant speed of the robot end effector, particularly when the robot arm is in a complex folded configuration, thus restricting the power delivered by the cue and ultimately the range of possible cue forces. Finally, commanding the robot arm to make a move at speed is potentially dangerous to both the machine itself and any nearby equipment or person in case any fault in the software or hardware manifests itself.

As a result, it is neither practical nor safe to make the robot arm emulate the human cueing action. The key aspect of taking a shot is the motion of the cue at speed hitting the cue ball, thereby delivering an impulsive blow upon it. By utilising a pneumatic cylinder as the snooker cue, this action can be reproduced. With such a device, none of the problems associated with taking a shot by physically moving the robot arm exists anymore. This is because by careful selection of the positions at which the PUMA is located by the SKF linear driver, the robot is able to position the pneumatic cylinder at the cue ball anywhere on the snooker table in any shot direction. In addition, no robot motion is involved in the actual physical mechanism of shot taking, which eliminates the danger associated with a fast moving robot arm.

Pneumatic cylinders come in all sort of specifications and sizes to suit a wide range of industrial applications. In this application, the pneumatic cylinder is to be attached to the PUMA robot arm and its payload of 5 lbs. must not be exceeded. This places a size constraint on the type of pneumatic cylinder that can be used. After surveying the market for suitable cylinders that are readily available, a mini ISO cylinder of 25mm bore and 40mm stroke is chosen. It is a double acting, double cushioned cylinder that offers good efficiency and control. An internally threaded aluminium cap is machined to screw onto the piston rod and a genuine snooker cue tip is glued to the tip of the cap. The pneumatic cylinder after conversion is shown in figure 1.4.

With the cue tip fitted, the pneumatic cylinder is converted into a pneumatic snooker cue. To take a shot, the robot arm would line up the pneumatic snooker cue at the cue ball pointing in the required direction, with the correct amount of tilting if necessary. A Shrader Bellows V.I.P. pressure control unit is installed to control the power of the shot. The V.I.P. pressure control unit sets the pressure directed to the outlet port of the pneumatic cue according to a voltage input. This input voltage (dc) ranges from 1 V (minimum output pressure) to 5 V (maximum). This input voltage is derived from the status of 8 of the 32 PUMA output lines. In other words, there are 256 digitally controlled levels of cue force. The firing of the pneumatic cue is triggered on receiving a digital signal from the PUMA controller whereby pressurised air at 120 p.s.i. is directed to the inlet port of the pneumatic cue. This variability and controllability of pneumatic cue force is a key aspect of strategic snooker play and the cue force control mechanism is explained in details in chapter 2.

1.6 Conclusions

The case for the automation of skilled tasks traditionally carried out by human has been put forward. A brief description of the system hardware has also been presented. The function of each individual piece of equipment is covered and details of the necessary interaction and co-ordination of all system components is the subject of the following chapter.

The entire system software can be broken down into many logical modules e.g. image processing, visual servoing, shot planning, etc. Chapter 2 deals with the integration of the various pieces of equipment to form an integral unit. The behaviour of snooker balls in motion and collision is studied in chapter 3. The extraction of snooker ball position and colour information from a raw gray-scale image, together with the implementation of an eye-in-hand visual servoing module are discussed in chapter 4. Algorithms for identifying feasible shots and comparing the relative difficulties of different shots can be found in chapter 5.

A strategy for analysing the game of snooker is devised which allows the snooker game plan to be represented as a game tree is presented in chapter 6. From the snooker game tree, a 1-step ahead game plan can be devised. The game plan includes a (target ball, pocket, cue force) tuple that specifies the first shot to play and the anticipated second shot. The search for the optimal plan is guided by the combined difficulty of the first and second shots. Once a (target ball, pocket, cue force) tuple is generated, the relevant robot motion commands are issued to physically take the shot.

Accuracy of the snooker ball dynamics model is the key to the success of the system. A technique known as Genetic Algorithms is used to determine the correct values of the various ball motion parameters. Genetic algorithms work in a way that mirrors biological evolution and are particularly powerful in dealing with search and optimisation problems. An original method is devised to assess the correctness of a mathematical model of snooker ball motion and collision. The result of the assessment is then fed into a genetic algorithm such that a better model will evolve, details of which can be found in chapter 7. Conclusions and future research work are discussed in chapter 8 and 9.

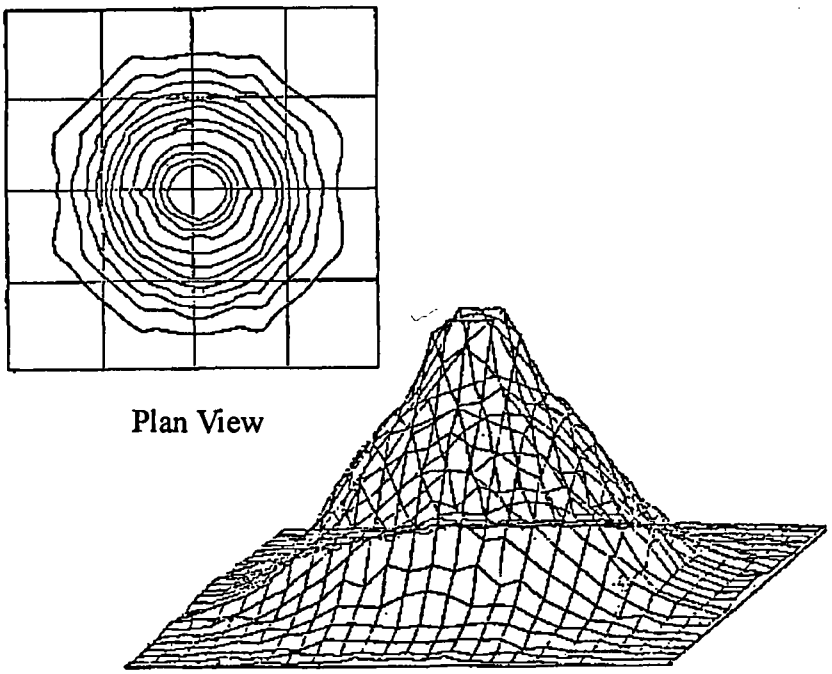


Figure 1.1 Luminance Intensity Distribution of a Point Light Source

Figure 1.2 Overhead Lighting Grid

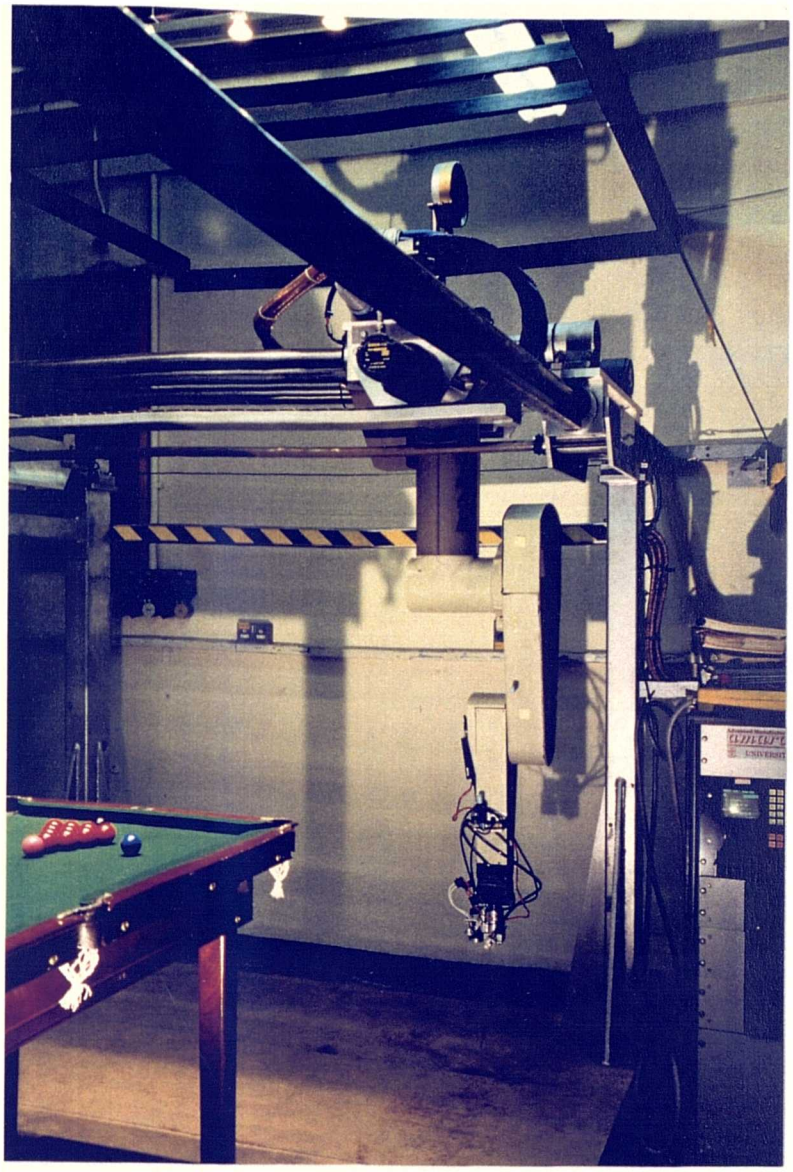


Figure 1.3 Upside-Down Installation of PUMA Arm to the SKF Linear Driver

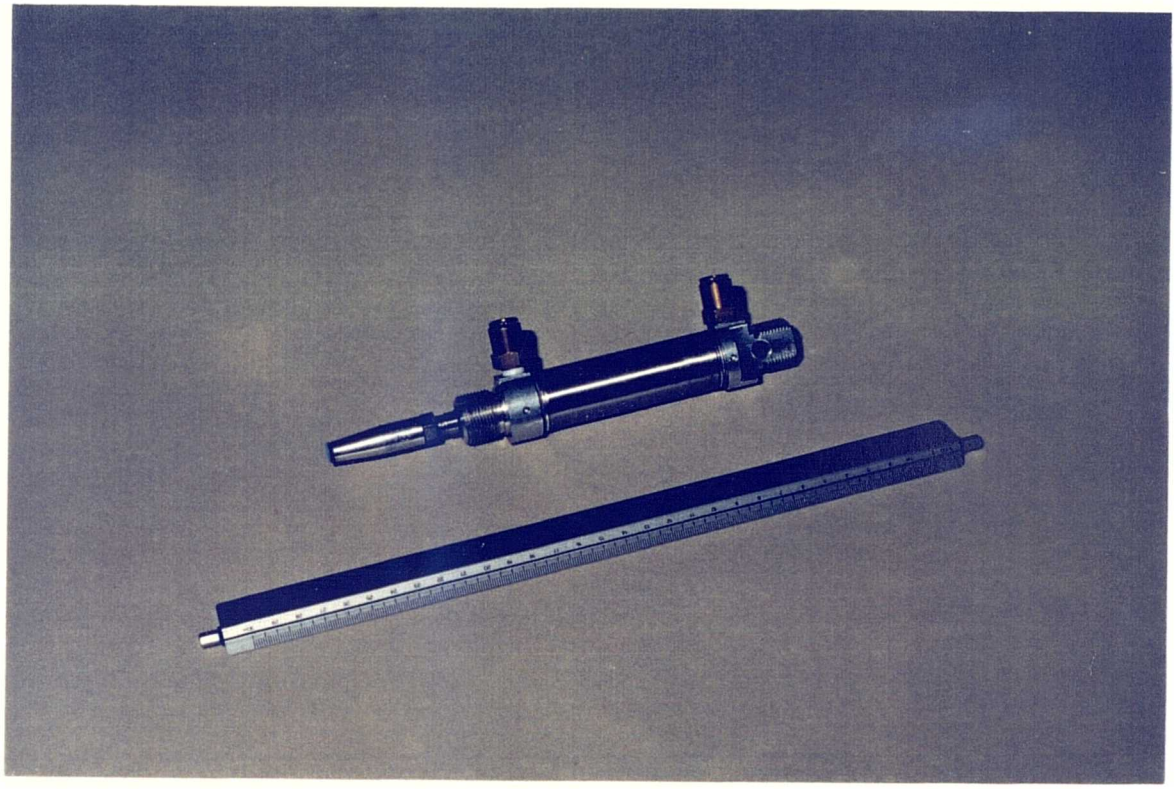


Figure 1.4 Converted Pneumatic Snooker Cue

Chapter 2 System Communication, Integration, and Calibration

Having identified the individual system components in the previous chapter, the way in which they are made to co-operate and work as an integral system will be discussed. The system architecture is depicted in figure 2.1. Although the figure does not contain every system detail, the major modules and flow of information are clearly illustrated. A 3-levels system hierarchy is established such that the Automatix AV-5 sits at the top with the PUMA robot arm under its control. At the middle level, the PUMA controller in turn controls both the SKF Linear Driver and the pneumatic cue at the bottom level. The various interfaces are discussed in detail below.

2.1 Automatix-PUMA Interface

The Automatix AV-5 image processor serves two purposes in the system. Apart from its obvious role of performing image processing, it also acts as the central processor of the entire system. Positional and colour information of all snooker balls are computed and stored in RAIL data structures. Shot planning and other associated software are also resident in the AV-5. This arrangement would minimise the amount of data that need to be transferred and only the essential parameters defining a shot i.e. cue ball position, cue orientation and cue force have to be sent from the AV-5 to the PUMA controller. This decision is supported by the fact that RAIL 6.04 is a more powerful and versatile programming language than VAL-II. The SKF Linear Driver and pneumatics control are transparent to the Automatix AV-5.

The AV-5 supports a single RS-232 serial communication port but the PUMA controller supports none. To set up a communication link between the two devices, the communication interface scheme recommended in the PUMA technical manual is adopted. This involves connecting the RS-232 port of the AV-5 to the Terminal port of the PUMA using a serial data transmission cable. With regard to the data transmission cable, a 25-pin male RS-232 'D' connectors is fitted at the AV-5 end while a 10-pin male cannon connector (conforming to BS 9522 F0017) is fitted at the PUMA end. Figure 2.2 illustrates the wiring of the data transmission cable and figure 2.3 shows the data transmission protocol.

With this data link established, the PUMA controller is ready to receive and treat any command transmitted from the AV-5 as if it were sent from the PUMA console. On the AV-5 end, however, a software module is written to deal with data communication with the PUMA. This module has two major functions : 1. to transmit VAL-II commands and receive any echo generated by the PUMA, and 2. to ensure correct data transmission. The second function is of particular importance because all robot positions and orientations are computed by the AV-5, then sent to the PUMA for execution. Any corrupted data could result in the PUMA controller driving the PUMA robot arm to an unexpected position and orientation which can be extremely hazardous.

The AV-5 controls the PUMA in a general pattern of first sending a set of program parameters and then initiating execution of a VAL-II program. Indeed the AV-5 is to command the PUMA to perform 1 of 5 possible actions :

(1) to position the On-board Camera over a snooker ball to start visual servoing [2]

(2) to shift the on-board camera in the X-Y plane to locate a ball accurately [2]

(3) to set the pneumatic cue force [1]

(4) to line up the pneumatic cue at the cue ball to take a shot [7]

(5) to drive the robot arm back to the home position [0].

The number of parameters required in each action is bracketed above. All parameters are real numbers except the cue force value which is an integer. Consequently, any VAL-II command sent to the PUMA from the AV-5 is either a variable assignment statement such as 'do x = 1' where the variable 'x' is assigned an integer value of 1, or a program execution statement such as 'ex prog' whereby a VAL-II program named 'prog' is run.

As with any sort of data transmission, corruption of data is a possibility. Safety is a prime consideration in any system involving robots and the data defining the position and orientation of the robot arm must be transmitted and received correctly. There are two levels of protection against corruption of data sent from the AV-5 to the PUMA. On receiving a command, the PUMA controller echoes the incoming character string back to the AV-5, in exactly the same way as a key typed at a computer keyboard is echoed on the monitor.

The first level of protection is thus achieved by comparing the echo with the original command, which is taken care of by the RAIL function 'PUMA_COMMAND'. For example, the RAIL function call

```
PUMA_COMMAND ('ex prog')
```

sends the character string 'ex prog' to the PUMA controller. On receiving this command, the PUMA controller echoes the character string back to the AV-5.

From the point of view of the AV-5, transmission of a command is repeated until the incoming echo string matches the out-going command string. This mechanism is able to detect most but not all cases of data corruption. It fails when a command is corrupted during transmission to the PUMA. Normally, this corrupted command is echoed back to the AV-5 unchanged and thus detected. Unfortunately, it is not impossible for the echo to be corrupted to become the original command during transmission back to the AV-5, thereby making the data transmission appear as error-free, as shown in figure 2.4. It has to be acknowledged here that the chance of this 'reciprocating' data corruption occurring is very small, but nonetheless possible. Note that if this type of data corruption happens to any non-numerical characters in a command, no action would result because the command becomes incomprehensible to the PUMA controller. Also, all 5 VAL-II programs that the AV-5 may execute have program names of different lengths and there is no possibility of mistaking one with another.

The worst scenario of data corruption that escapes detection by the first level of protection is that of a number becoming altered during transmission. At best the robot could be driven to an unintended position. At worst, damage to human or machines could result. To prevent this type of data corruption from actually causing any damage, a classic fail-safe mechanism is employed. Whenever a variable assignment statement has to be transmitted to the PUMA, it actually gets transmitted three times where the variable identifier in the statement is suffixed by '1', '2', and '3'.

The RAIL function 'PUMA_SAFE_SEND' performs this operation of triplicating a variable assignment statement and sending them to the PUMA controller.

For example, the RAIL function call

```
PUMA_SAFE_SEND ( x , 1 )
```

where the first argument is the variable name and the second the value to be assigned to it, will result in 3 RAIL function calls

```
PUMA_COMMAND ('do x1 = 1')
PUMA_COMMAND ('do x2 = 1')
PUMA_COMMAND ('do x3 = 1')
```

Bearing in mind that a set of variable assignment statements is always followed by a program execution statement, it is possible for each program to check the consistency of the parameters received. The program prints an error message and halts execution if all three copies of a parameter differ, otherwise execution proceeds as normal. If all three copies of the parameter received are identical, then it is assumed that no data corruption has occurred. If one copy is different from the other two, then that copy is treated as corrupted data and discarded. In practice, no VAL-II program has ever halted execution because of corrupted parameters received from the AV-5, which illustrates the general reliability of the communication link. Admittedly, by sending 3 commands instead of 1, transmission time will be trebled plus some overhead at the PUMA end to check if they agree. But at a transmission rate of 9600 Baud, the safety benefits far outweigh the additional computation costs.

As a summary, the Robot Snooker Player System only needs two functions to command the PUMA :

`PUMA_SAFE_SEND ( var, n )`

where `'var'` is a parameter identifier and `'n'` is the value assigned to it, and

`PUMA_COMMAND ('ex prog')`

where `'prog'` is a PUMA VAL-II program name.

2.2 PUMA-SKF Interface

Although the SKF Linear Driver acts as a slave device to the PUMA, its function is vital to the system. Without the SKF, the range of tasks that can be performed by the PUMA robot arm will be severely limited by the PUMA work envelope. Indeed it is a common practice to have robots mounted on tracks or gantries to enhance their abilities through the provision of locomotion. In the work of Su and Zheng **[Su91]**, the workspace of a robot is expanded through the use of a legged mobile platform, which essentially serves the same purpose as the SKF Linear Driver.

In this application, the PUMA robot arm is required to perform 2 tasks : to position the on-board camera over a snooker ball to perform visual servoing and to line up the pneumatic cue at the cue ball for shot taking. The orientation of the on-board camera is fixed during visual servoing but the orientation of the cue depends on the shot angle. In both cases, the robot tools i.e. the on-board camera and the pneumatic cue must be able to reach any ball anywhere on the snooker table. In addition, the robot always attempts to line up the cue horizontally but if any obstacle e.g. a nearby ball or cushion is detected, the minimum amount of cue tilting will be introduced to clear the obstacles.

The work envelope of a basic PUMA robot arm is shown in figure 2.5, within which the intersection point of the PUMA joint axes 4, 5, and 6 (the centre point of the robot wrist) can be positioned. The entire envelope is not really relevant to this application which requires the PUMA to work in two horizontal planes. The first horizontal plane is defined by the operating height of the on-board camera and the second is defined by the centres of the snooker balls on the snooker table.

A method is devised to generate the work envelope of the PUMA with respect to the on-board camera and the pneumatic cue. Details of the method can be found in Appendix A. In general, the PUMA robot arm can reach any point within its work envelope in a maximum of 8 postures, depending on (1) the position of the elbow (BELOW or ABOVE), (2) the sense of the folded arm (LEFTY or RIGHTY), and (3) the state of the wrist (FLIP or NOFLIP). Factor (3), the state of the wrist, does not matter in this application and thus the number of possible arm postures is reduced to 4 : { [BELOW, LEFTY], [BELOW, RIGHTY], [ABOVE, LEFTY], [ABOVE, RIGHTY] }. Figure 2.6 shows the PUMA work envelopes with respect to the pneumatic cue in these 4 arm postures and figure 2.7 shows the PUMA work envelopes with respect to the on-board camera, also in the 4 arm postures.

Figure 2.6 and 2.7 show that with respect to either the on-board camera or the pneumatic cue, the work envelopes corresponding to the postures [ABOVE, LEFTY] and [ABOVE, RIGHTY] are much smaller than those corresponding to the other two postures. Therefore, a decision is made here NOT to operate the PUMA robot arm in the [ABOVE, LEFTY] and [ABOVE, RIGHTY] postures without compromising the robot's work envelope.

Figure 2.8 shows the resultant pneumatic cue work envelope with the arm in the [BELOW, LEFTY] and [BELOW, RIGHTY] postures which is approximated by a circle with a hollow centre, a DISK, of inner diameter 320 mm and outer diameter 590 mm. The area inside the inner circle and the area outside the outer circle are inaccessible by the robot arm. Figure 2.9 shows that DISK which approximates the pneumatic cue envelope also lies completely within the combined on-board camera work envelopes in the [BELOW, LEFTY] and [BELOW, RIGHTY] postures. Hence DISK represents the union of the pneumatic cue and on-board camera work envelopes.

It now follows that if a set of SKF base points $\{P_1, P_2, P_3, \ldots P_n\}$, where the PUMA base can be driven to, are defined such that the corresponding work envelopes $\{DISK_1, DISK_2, DISK_3, \ldots DISK_n\}$ cover the entire snooker table, then the PUMA robot arm will be able to position the pneumatic cue at the cue ball lying anywhere on the table in any direction or carry out visual servoing on any snooker ball.

There are many possible arrangements to have n DISKs covering the snooker table. After much consideration, an incremental approach is devised requiring 7 SKF positions for system operation plus a 'HOME' position for parking the PUMA and SKF when the system is not in use. This happens to be very convenient because the PUMA specifies which SKF position to go to via a set of output switches, each of which can be 'on' or 'off'. By using 3 PUMA output switches, any 1 of the 8 SKF positions can be specified.

The area of the snooker table play area (i.e. that defined by and not including the table cushions) measures 720 mm by 1550 mm. Figure 2.10 (a) is a scale drawing showing this play area and the 7 SKF base points at which the centre of the base of the PUMA robot arm can be positioned. Figure 2.10 (b) shows how the snooker table play area is covered by the resultant extended PUMA work envelope. Note that in figure 2.10 (a), the 7 SKF base points are defined relative to the snooker table cushions. Their actual co-ordinates in both the SKF and world co-ordinates system will be defined in chapter 2.4.

An algorithm is designed and implemented which takes as input a general point (x, y) in world co-ordinates and outputs the required SKF position i and the target point (X_PUMA, Y_PUMA) in the PUMA co-ordinates at SKF base point i, if a solution exists. Otherwise the point (x, y) is beyond the reach of the PUMA robot arm. Strategic definition of the 7 SKF base points ensures that any point on the snooker table can be reached. Figure 2.11 is a set of pseudo-codes that outlines the algorithm, which is graphically explained in figure 2.12. Referring to figure 2.10 (b), the algorithm is equivalent to scanning the PUMA work envelopes starting with DISK1 and checking if the point (x, y) lies within the envelope. If it does not, DISK2 will be tested and so on until a solution is found or all SKF positions are exhausted.

Suppose a point (x, y) on the snooker table is to be reached (it does not matter whether it is a cueing or visual servoing operation because DISK is the union of the on-board camera and pneumatic cue envelopes), a VAL-II program will scan the SKF positions from the 1^{st} to the 7^{th}. For SKF position i, the corresponding PUMA base position (which is the origin of the PUMA co-ordinate system) in world co-ordinates is (XBase[i], YBase[i]) with work envelope $DISK_i$. It then follows that if d_i, the distance between (x, y) and (XBase[i], YBase[i]), is less than 590 mm, then the PUMA robot can be driven by the SKF to the i^{th} position to reach point (x, y). This involves converting the point (x, y) from the world co-ordinates system into the $PUMA_i$ co-ordinates system. The $PUMA_i$ co-ordinates system is that of the PUMA at SKF base point i, as seen in figure 2.12. In other word, d_i is the magnitude of the vector $(x\text{-}XBase[i], y\text{-}YBase[i])^T$.

It is not necessary to test whether d_i is less than 320 mm, in which case (x, y) lies inside the inner circle of $DISK_i$ and is inaccessible, provided that the target point (x, y) is inside the snooker table. This is a direct result of the incremental nature of the strategy which would have identified the $(i\text{-}1)^{th}$ SKF base point as the solution without the need to test the i^{th} SKF position. Another side-effect of this strategy is that the PUMA is always driven to a point with negative PUMA X co-ordinate, therefore the PUMA needs only operate in the [BELOW, RIGHTY] posture to reach any point on the snooker table, see figure 2.6 and 2.7.

The observant reader would have spotted in figure 2.10 (b) that $DISK_4$, $DISK_5$, $DISK_6$, and $DISK_7$ together cover the entire rectangle, meaning that just 4 SKF points are sufficient instead of 7. It is certainly feasible to use 4 SKF base points, as shown in figure 2.13 but that would mean that sometimes the arm posture will have to change from [BELOW, LEFTY] to [BELOW, RIGHTY] or vice versa.

When such a change is required, the PUMA has to be driven by the SKF to the 'HOME' position so that the posture changing arm motion will not hit the snooker table. In the author's opinion, it is more reassuring to operate the PUMA is a single posture, with regard to both safety and efficiency, hence 7 SKF base points (plus the 'HOME' position) are defined where the PUMA robot can be positioned. Also the workload in terms of machine movements is better shared between the SKF Linear Driver and the PUMA using 7 SKF points instead of 4.

After the PUMA program has determined which one of the seven SKF base points the PUMA robot is to be positioned at to reach a general point (x, y) in world co-ordinates, this piece of information has to be transmitted to the SKF Linear Driver. This process is divided into 2 sub-tasks : (1) transmission of the positional information through 3 binary communication lines and (2) transmission of a clock pulse to start SKF motion.

Both the SKF and the PUMA controller support a number of binary input and output signals. To allow the PUMA to control the SKF, 4 PUMA output signals have to be connected to 4 SKF input signals. Three of the four signals are used to specify the SKF position number which is an integer ranging between 0 and 7. Point 0 is the PUMA and SKF parking position 'HOME'. The remaining signal is used to send a clock pulse which triggers the actual SKF motion. Figure 2.14 shows how the PUMA and SKF communication lines are connected.

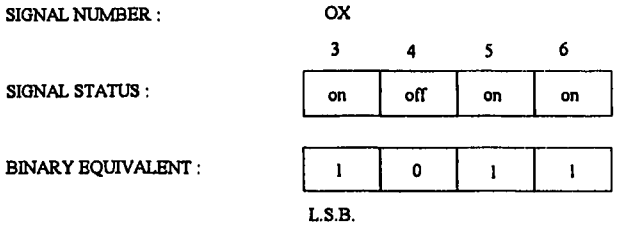
VAL-II provides a very useful function 'PC' which sets the value of a number of output lines according to an integer parameter. The syntax of this instruction is :

```
PC <signal> , ( <bits> ) = <value>
```

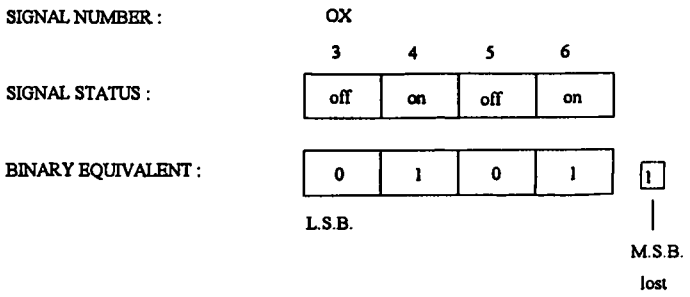
The <signal> argument specifies the signal associated with the least significant bit of the binary representation of <value>. The <bits> argument specifies how many signals are to be set, starting at <signal> and running consecutively higher. For example, execution of the VAL-II instruction :

```
PC 3,4 = 13
```

results in the following PUMA output signals settings :



where '1101' is the binary representation of the decimal integer '13'. Note also that if insufficient number of signals are allocated, the more significant bits will be lost. In the above example, if '13' is replaced by '26', which in binary form is '11010', the signal status becomes :



According to figure 2.14, by issuing the instruction

```
PC 13,3 = i
```

$i = \{0,1,2,3,4,5,6,7\}$

where i is an integer specifying which SKF position the PUMA robot is to be positioned at, the PUMA output signals OX13, OX14, and OX15 will be set to represent i in binary form. The corresponding SKF input lines E7, E8, and E9 will be set as the binary representation of i. Note that the SKF does not initiate the required move until a pulse signal is received from the PUMA controller.

To create a pulse signal, PUMA output signal OX16, which is dedicated as a trigger, is set to 'off' and then 'on' by the following VAL-II instructions :

```
SIGNAL -16
WAIT
SIGNAL 16
```

A 'WAIT' instruction introduces a delay of 28 millisecond to ensure there is sufficient time for the first 'SIGNAL' instruction to propagate to the SKF processor. This clocking mechanism is illustrated in figure 2.15.

On receiving the clock pulse through input signal E10, the SKF program (number 100, all SKF programs are referenced by numbers) decodes the positional information using a series of test and jump instruction to determine which of the 8 pre-defined locations is specified and then starts the move. After a move is made, the program loops back to the beginning and waits for the next clock pulse from the PUMA controller, ready for another move.

The objective as defined at the beginning of this section is thus met by extending the PUMA work envelope by physically moving the PUMA robot arm to various specific points using the SKF Linear Driver.

2.3 PUMA-Pneumatics Control Interface

The pneumatic cue plays a very important role in the system. Not only does it perform the physical task of striking the cue ball but more significantly, it is through the control of cue force that positional play is achieved. Therefore, the system must be able to control the cue force by regulating the supply air pressure to the pneumatic cue, both accurately and reliably.

The V.I.P pressure regulator regulates the output air pressure proportionally according to an input control signal in the form of a variable voltage (dc) which ranges from 1V to 5V. The V.I.P. is factory calibrated such that when the input voltage equals 1V, the output air pressure equals zero while at 5V, the output air pressure is equal to 100 p.s.i., which is the maximum supply air pressure it can withstand. Because of the binary nature of the PUMA output signals (each signal is either 'on' or 'off') a D.A.C. (Digital-to-Analogue Converter) is required to convert a set of binary signals into an analogue dc voltage within the V.I.P.'s input range of 1 to 5V. The total number of V.I.P. output pressure levels available is a function of the number of PUMA output signals allocated. For instance, n PUMA output signals allocated to the D.A.C. would provide 2^n user selectable V.I.P. output pressure levels.

In this set-up, an air compressor maintains a supply of pressurised air at 120 p.s.i. to the V.I.P. regulator and a solenoid valve. Since the maximum supply pressure to the V.I.P. cannot exceed 100 p.s.i., a pressure regulator is fitted between the air compressor and the V.I.P. to reduce the pressure to 100 p.s.i..

The output from the V.I.P. is then connected to the outlet port of the pneumatic cue which causes the piston rod to retreat into the cylinder in the presence of compressed air. The output of the solenoid valve is connected to the inlet port of the pneumatic cue. The solenoid valve is normally closed i.e. output pressure equals zero and therefore the pneumatic cue is normally in the 'cue-in' state. When the valve is open, compressed air at 120 p.s.i. is directed to the inlet port of the pneumatic cue, causing the piston rod to extend from the cylinder, which is the action required to strike a ball. The control of the pneumatic cue force and actuation of the cue is shown in figure 2.16.

The V.I.P. supplies a variable cushion air pressure $P_{cushion}$ (0-100 p.s.i.) which acts against the fixed cue-out pressure of 120 p.s.i. when the solenoid valve is open (zero p.s.i. otherwise). Therefore the effective cueing air pressure equals (120 - $P_{cushion}$) p.s.i., ranging from 20 p.s.i. (when $P_{cushion}$ = 100 p.s.i.) to 120 p.s.i. (when $P_{cushion}$ = 0 p.s.i.).

Since 8 PUMA output signals are allocated to specify the cushion air pressure $P_{cushion}$, there are 256 levels of effective cueing air pressures. For simplicity, cue force level 1 is defined as the minimum level (maximum $P_{cushion}$) and cue force level 256 is the maximum (zero $P_{cushion}$). Bearing in mind the antagonistic relationship between $P_{cushion}$ and the effective cue force, the following VAL-II instruction sets PUMA output signals 1-8 according to the specified cue force variable 'cf' :

```
PC 1, 8 = 256-cf          cf = {1, 2, 3,..., 254, 255, 256}
```

The functioning of the VAL-II instruction 'PC' is already described in chapter 2.2. After the desired cue force is selected and set, the cue is ready to be triggered. The following VAL-II instruction :

```
SIGNAL 9
```

switches output signal number 9 to the 'on' state which opens the solenoid valve, as seen in figure 2.16. Once the cue is fired, it stays in the 'cue-out' state until the solenoid valve is closed by the instruction :

```
SIGNAL -9
```

that switches output signal number 9 to the 'off' state, allowing the pressure from the outlet port to push the piston back.

2.4 System Calibration

The overall performance of the robot snooker system is critically dependant on how accurately the system components are calibrated. Co-ordinate systems have to be defined and aligned, and the position of the PUMA robot in WORLD co-ordinates must be precisely known at all times.

The SKF Linear Driver is installed and set up so that the device is free to move in a plane parallel to the horizontal, as is the snooker table. The PUMA robot arm is installed upside-down to the SKF tool mounting plate such that the PUMA joint 1 axis is normal to the horizontal plane. Their respective co-ordinate systems are shown in figure 2.17.

2.4.1 Alignment of the SKF and PUMA Axes and Pneumatic Cue

As seen in figure 2.17, the SKF and PUMA axes are 'virtually' aligned with each other. However, because of machining tolerances of the robot base mounting bores and SKF tool mounting plate holes, there exist a rotational discrepancy of approximately 1 degree between the two co-ordinate systems, as in figure 2.18.

The SKF Linear Driver is a substantial piece of equipment and the orientation of its co-ordinate system cannot be changed except by re-installing it. On the other hand, the PUMA software supports a command 'BASE' which offsets and rotates the PUMA base co-ordinate system. As the Y-axis of either co-ordinate system is orthogonal to the X-axis, it is only necessary to align the PUMA X-axis with the SKF X-axis.

The SKF X-axis is 'recorded' by a piece of custom made tool, called the axis alignment tool. The tool is a piece of square-section steel tube with 6 steel rods (threaded at one end and pointed at the other) screwed into it, see figure 2.19. The idea is to clamp the axis alignment tool onto the snooker table so that tips of the 6 steel rods are roughly lined up with the SKF X-axis. The PUMA robot arm is then driven to a posture such that the pneumatic cue tip points at the steel rod tips. By driving the SKF in the X-direction and with the use of feeler-gauges, each of the 6 steel rods can be turned into or out of the square-section tube so that a fixed gap is maintained between the cue tip and each steel rod tip.

Once the 6 steel rods are lined up with the SKF X-axis, the tool will not be further adjusted. Having determined the SKF X-axis, the SKF is then held stationary, allowing the PUMA to be driven along its X-axis. As the tip of the pneumatic cue moves pass the tip of each of the 6 steel rods, the gap between is checked using feeler gauges. Figure 2.20 shows the 3 possible situations of axis alignment. A fixed gap is maintained if and only if the PUMA X-axis is aligned with the SKF X-axis, as in CASE 2. If, however, CASE 1 in figure 2.20 occurs, the VAL-II command

$$\text{BASE } ,,, \theta \qquad\qquad \theta > 0$$

is issued to rotate the PUMA co-ordinates system in the anti-clockwise direction about the Z-axis (viewing from the negative Z-axis) to align the PUMA and SKF X-axes. The tool does not measures the angular discrepancy between the X-axes. Therefore the value of θ has to be found by trial and error and repeating the alignment checking process until a fixed gap is maintained between the cue tip and the tips of the steel rods. If CASE 3 in figure 2.20 occurs, a negative θ is substituted into the command to rotate the PUMA co-ordinate system clockwise about the Z-axis. In both cases, the appropriate 'BASE' command is inserted into an initialisation program which sets up the PUMA co-ordinate system correctly when required, e.g. after the PUMA system has been powered-down.

To achieve accurate cueing, the orientation of the pneumatic cue has to be known precisely. It is decided that the pneumatic cue is to have the same orientation as the PUMA 'NULL' tool, which equals {0, 0, 0, 90, -90, 0} by default. The pneumatic cue tool definition is stored in a VAL-II location variable 'CUETOOL' as {170,0,80,90,-90,0}, the first 3 arguments indicate the displacement of the cue tool point from the 'NULL' tool point in the X, Y and Z directions and the last 3 arguments indicate that both tools have the same orientation. The 'NULL' and 'CUETOOL' are shown graphically in figure 2.21 together with the relative position of the pneumatic cue. To align the cue with the PUMA X-axis, the PUMA 'NULL' tool is first selected by the command 'TOOL'. Then the robot arm is driven to a point {x, y, z, 90, -90, 0} along-side the axis alignment tool as set above. The values of x, y, and z are irrelevant provided the pneumatic cue is level with the steel rods of the axis alignment tool. The PUMA can now be held stationary, and the SKF is then carefully driven to align the cue with the axis alignment tool, as in figure 2.22. Once the cue and the axis alignment tool are lined up, a securing bolt in the tool holder (which allows the cue to be removed or rotate about the Z-axis) is tightened and the cue is aligned.

2.4.2 Definition of Overhead Camera View Window and WORLD Co-ordinates System

Figure 2.23 shows the overhead camera view window and the WORLD co-ordinates system. The WORLD co-ordinates system is defined such that the WORLD X-axis and Y-axis are parallel to the corresponding axes of the overhead camera pixel co-ordinates system. Note that the overhead camera view window includes 20 mm of cushion on all 4 edges so that any snooker ball lying very close to a pocket is still within the view window. Conversion from pixel co-ordinates to WORLD co-ordinates is described in chapter 4.

2.4.3 Determination of SKF Linear Driver Step Size

Since the SKF Linear Driver does not support any encoder for positional feedback, positioning of the SKF relies on the counting of steps in the X and Y direction entirely. Using a LVDT, a fixed number of SKF steps can be translated into the resulting linear displacement in millimetres. This procedure is used to determine the SKF X and Y step-size, which are not necessarily equal. The SKF X-axis is aligned with but opposite to the WORLD X-axis while the SKF Y-axis is aligned with and runs in the same direction as the WORLD Y-axis.

The following test results were collected :

	150 SKF X-steps	150 SKF Y-steps
TRIAL 1	4.725 mm	4.742 mm
TRIAL 2	4.728 mm	4.742 mm
TRIAL 3	4.728 mm	4.745 mm
Mean	**4.727 mm**	**4.743 mm**

The linear displacements measured by the LVDT resulting from 150 X and Y steps are tabulated. To avoid rounding errors, the displacement resulting from a single X or Y step is not calculated. Instead, the linear displacement resulting from a single X-step is expressed as $\left(4.727/_{150}\right)$ mm and similarly a single Y-step equals $\left(4.743/_{150}\right)$ mm.

2.4.4 Definition of SKF Co-ordinate System Origin

Since the PUMA robot arm is mounted onto the SKF Linear Driver, its whereabouts in WORLD co-ordinates depends on the SKF position in terms of X and Y step numbers. The position of the SKF Linear Driver is defined as the origin of the PUMA co-ordinates system, which is the intersection point of the PUMA joint axis 1 and 2. To calibrate the SKF, and therefore the PUMA robot arm, it is to be positioned such that the WORLD Z-axis passes through the origin of the PUMA base co-ordinate system. The SKF co-ordinates at this position is set as (0, 0). Unfortunately, the actual PUMA origin is physically inside joint 1 of the robot which is inaccessible, and so it is impossible to directly aligned the WORLD origin with the PUMA origin.

To solve the problem, 2 PUMA locations with respect to the ' CUETOOL ' have to be defined. The first point P_1 is defined such that the cue lies horizontally in the PUMA X-Z plane (i.e. y = 0, orientation = {90, -90, 0}) just above the snooker table cushion. Thus positioned, the cue can be viewed as a vertical translation of the PUMA X-axis within the X-Z plane down to the snooker table level. The second point P_2 is defined such that the cue lies horizontally in the PUMA Y-Z plane (i.e. x = 0, orientation = {180, -90, 0}) just above the snooker table cushion. With the PUMA positioned at P_1 and P_2, the PUMA X and Y axes are effectively translated from their real position (which are level with the intersection between joint axes 1 and 2) down to the snooker table level. Referring back to figure 2.23, the WORLD X-axis lies parallel to the top cushion 20 mm above it and the Y-axis lies parallel to the left cushion 20 mm to the left of it. To calibrate the SKF Y-axis, the PUMA is to be positioned at P_1 (by manually driving the SKF) with the cue aligned with the WORLD X-axis (which is 20 mm above the top cushion as in figure 2.23). Once the cue is correctly positioned, the SKF Y co-ordinate is set to zero, thus defining the SKF Y-axis. The SKF X-axis is similarly defined using the point P_2. After the SKF X and Y axes have been set, the PUMA base co-ordinate system origin will be directly above the WORLD origin when the SKF is at (0, 0).

2.4.5 Definition of SKF Base Points

With the SKF co-ordinate system defined to coincide with the WORLD co-ordinate system, it is relatively straightforward to translate the SKF base points as defined in figure 2.10 (a) into their respective SKF co-ordinates. The only complications are the fact that the direction of the SKF X-axis runs opposite to the WORLD X-axis. Therefore all X co-ordinates of the SKF base positions have to be negated when they are converted from WORLD into SKF co-ordinates. This conversion is summed up in figure 2.24 which results in two sets of 8 (x, y) co-ordinates : 1 set in WORLD co-ordinates and the other in SKF co-ordinates. Note that the SKF can only move in complete steps and so all SKF co-ordinates must be rounded to their nearest integer values. Essentially the two sets of points are geometrically identical. The set of base points in WORLD co-ordinates is stored in a data structure in VAL-II which is required to run the incremental algorithm as described in chapter 2.2. When the PUMA controller has decided which SKF base point the robot is to be located at, the base point number is sent to the SKF controller. The SKF controller then looks up from its set of base points the position corresponding to the number received and makes the required move.

2.5 Conclusions

Integration of the system components described in chapter 1.5 has been achieved successfully. This involves designing a system architecture (see figure 2.1) and establishing the necessary interfaces. The Automatix AV-5 image processor has been chosen to act as the host computer within the intelligent robot snooker player system. The PUMA robot arm is under the control of the AV-5 at all times. In fact all devices within the system communicates in a strict one-to-one manner. An example is that the SKF Linear Driver is controlled by the PUMA but not by the AV-5 despite the fact that the AV-5 is at the top of the system hierarchy.

The nature of the system requires the PUMA robot arm (of substantial weight) to be transported within the workcell by the SKF Linear Driver. Also, the robot arm itself is capable of moving at very high speed. Any malfunction could have disastrous consequences. Hence safety features are built into the system communication to prevent possible malfunctions caused by data transmission errors.

Enlargement of the effective PUMA work envelope through the use of the SKF Linear Driver has also been discussed. The objective of ensuring that the PUMA robot arm can reach any ball anywhere on the snooker table is achieved by defining a set of locations where the PUMA can be positioned by the SKF. The SKF Linear Driver is under direct control of the PUMA controller. The PUMA controller also controls the pneumatic cue which is a vital component for strategic play in snooker. Cue force control is achieved through the regulation of the air pressure applied to the outlet port of the pneumatic cue. The pneumatic cue force is specified by the settings of 8 switches, resulting in 256 possible force levels.

At this point, the system is capable of positioning the cue or the on-board camera at any position on the snooker table. Cue force can be control but the system does not know how to make use of it yet. To have a functional snooker playing system, it must possess the abilities to see, to analyse the image, to comprehend the behaviour of snooker ball motion, and to make decisions. The following chapters will deal with these issues.

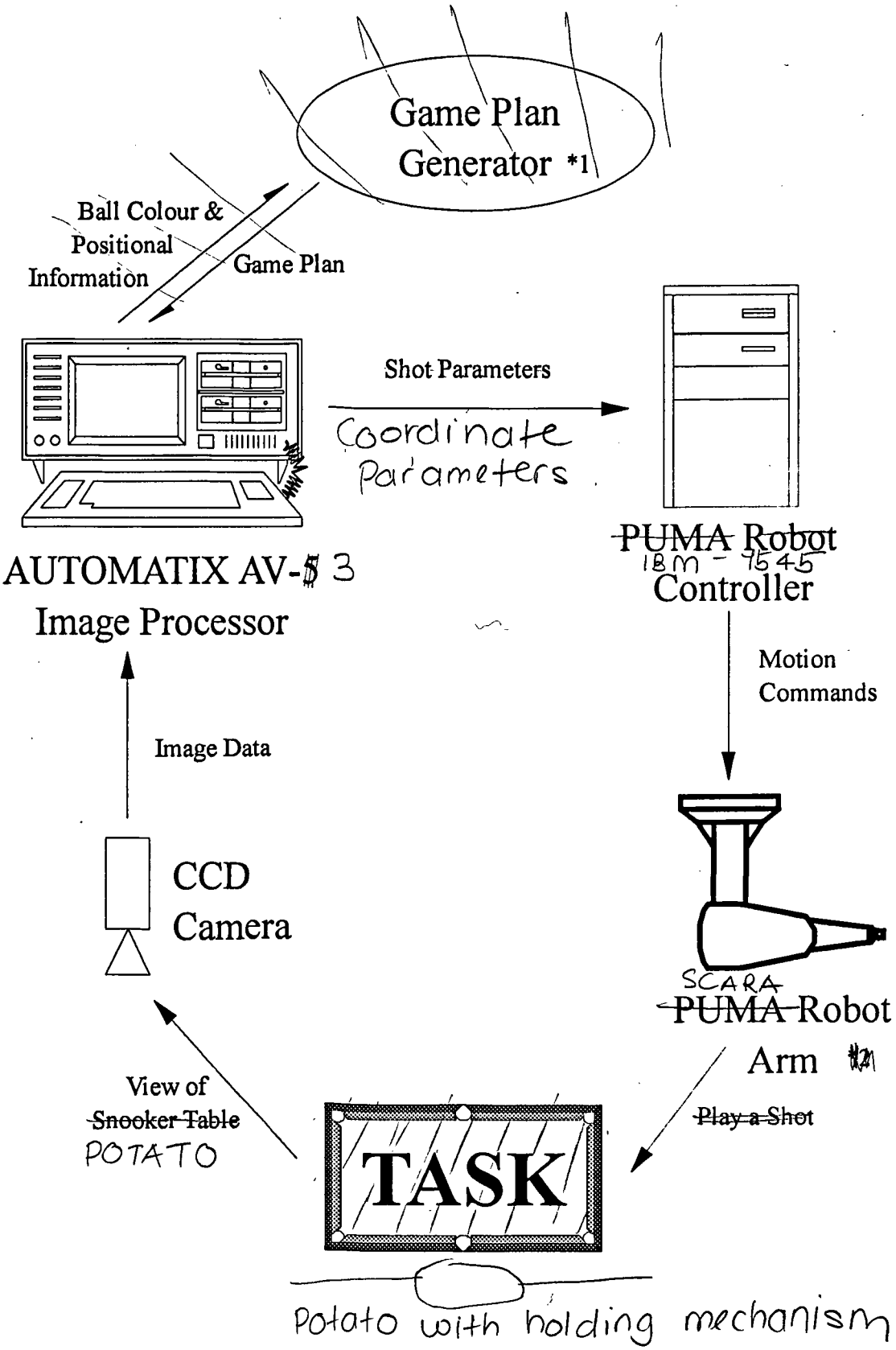


Figure 2.1 System Architecture

*1 This is a software module within the AV-5.

*2 The SKF Linear Driver acts as a slave device to the PUMA and is not shown in this diagram.

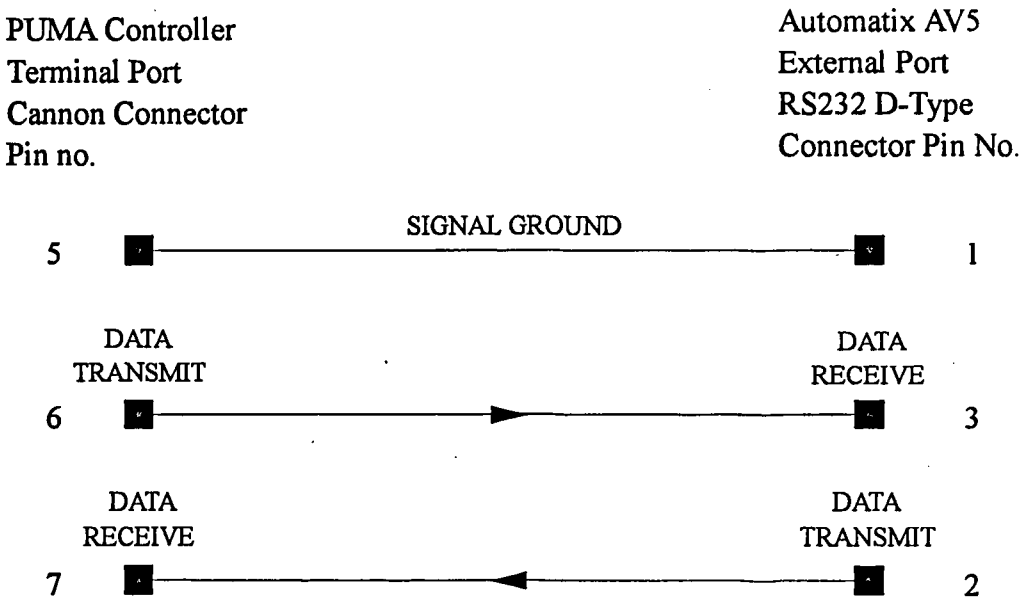


Figure 2.2 Serial Data Link Between PUMA and AUTOMATIX

Type of Transmission :	Asynchronous: 8 bit ASCII
Baud Rate :	9600
Mode :	Full Duplex
Stop Bit :	One stop bit
Parity :	None

Figure 2.3 Communication Protocol Between PUMA and AUTOMATIX

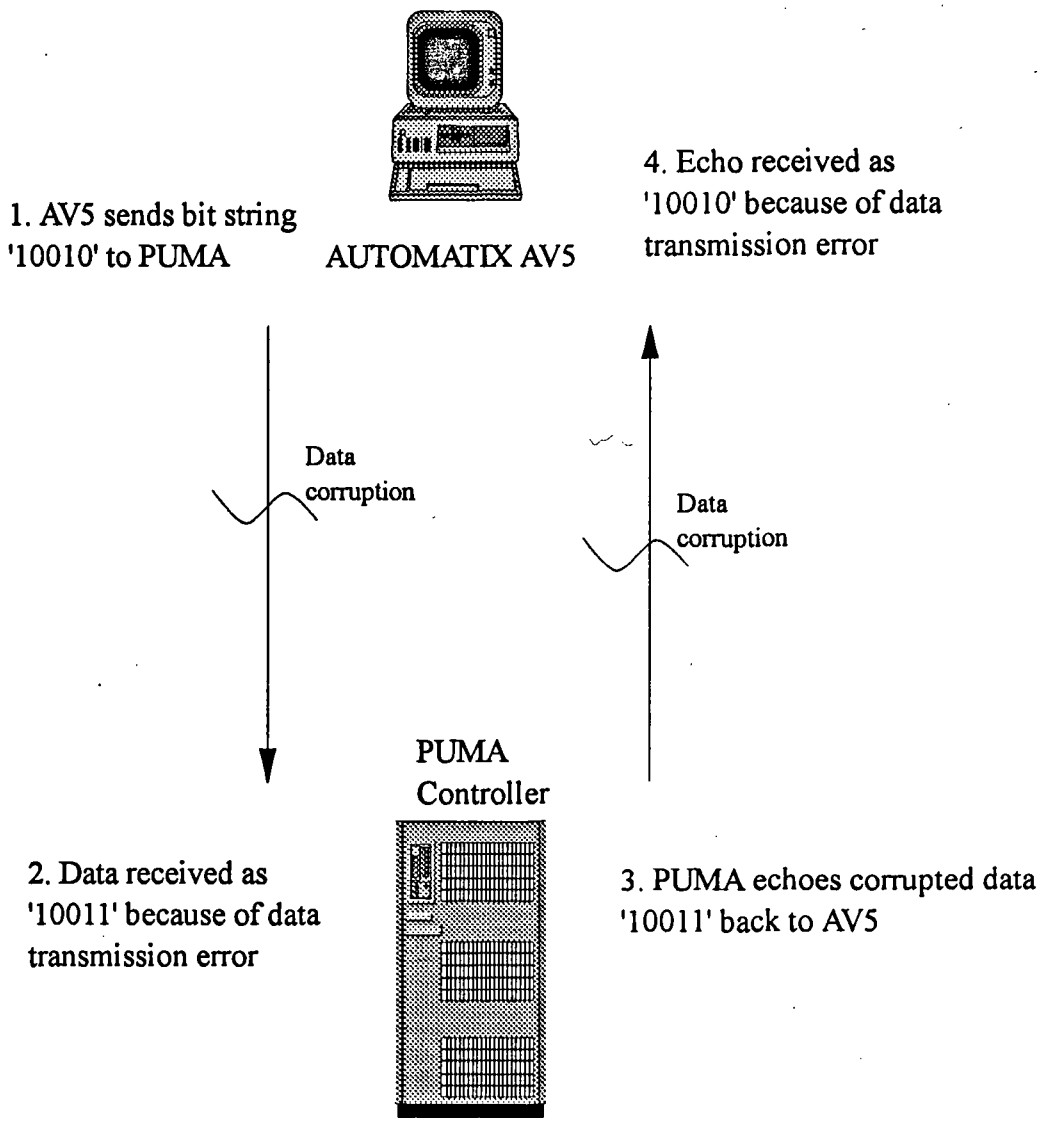


Figure 2.4 Example of Undetected Data Transmission Error

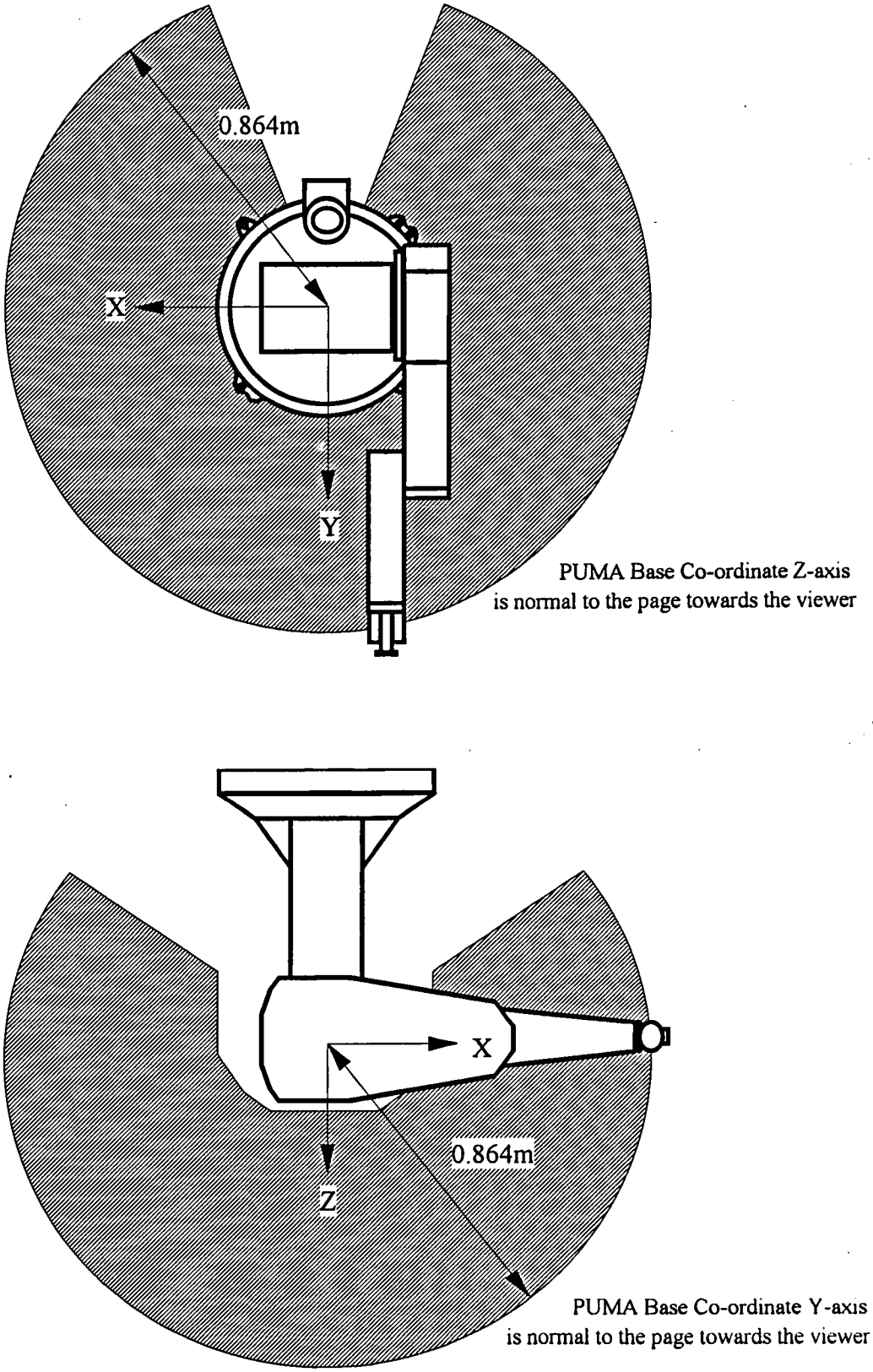
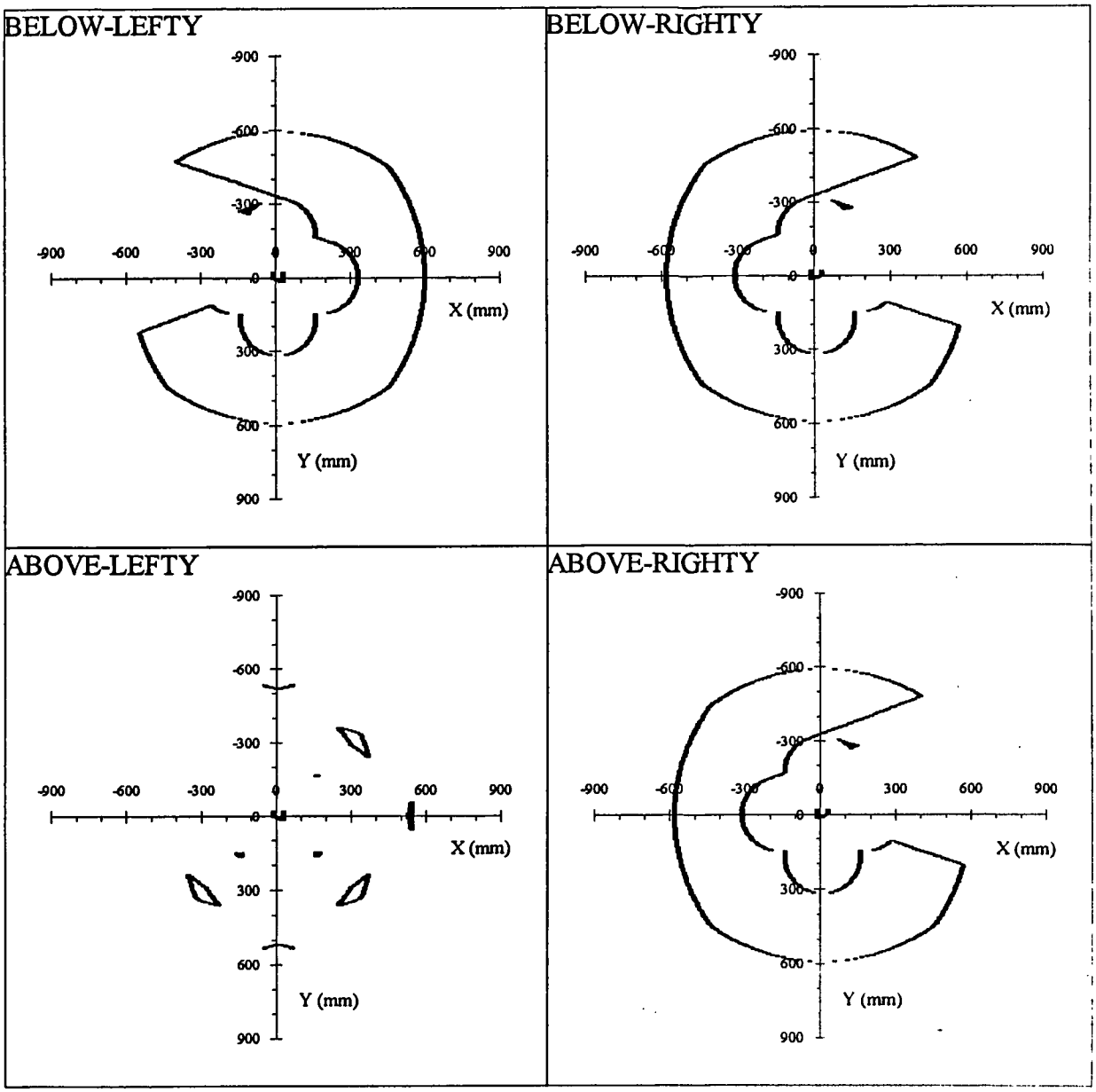
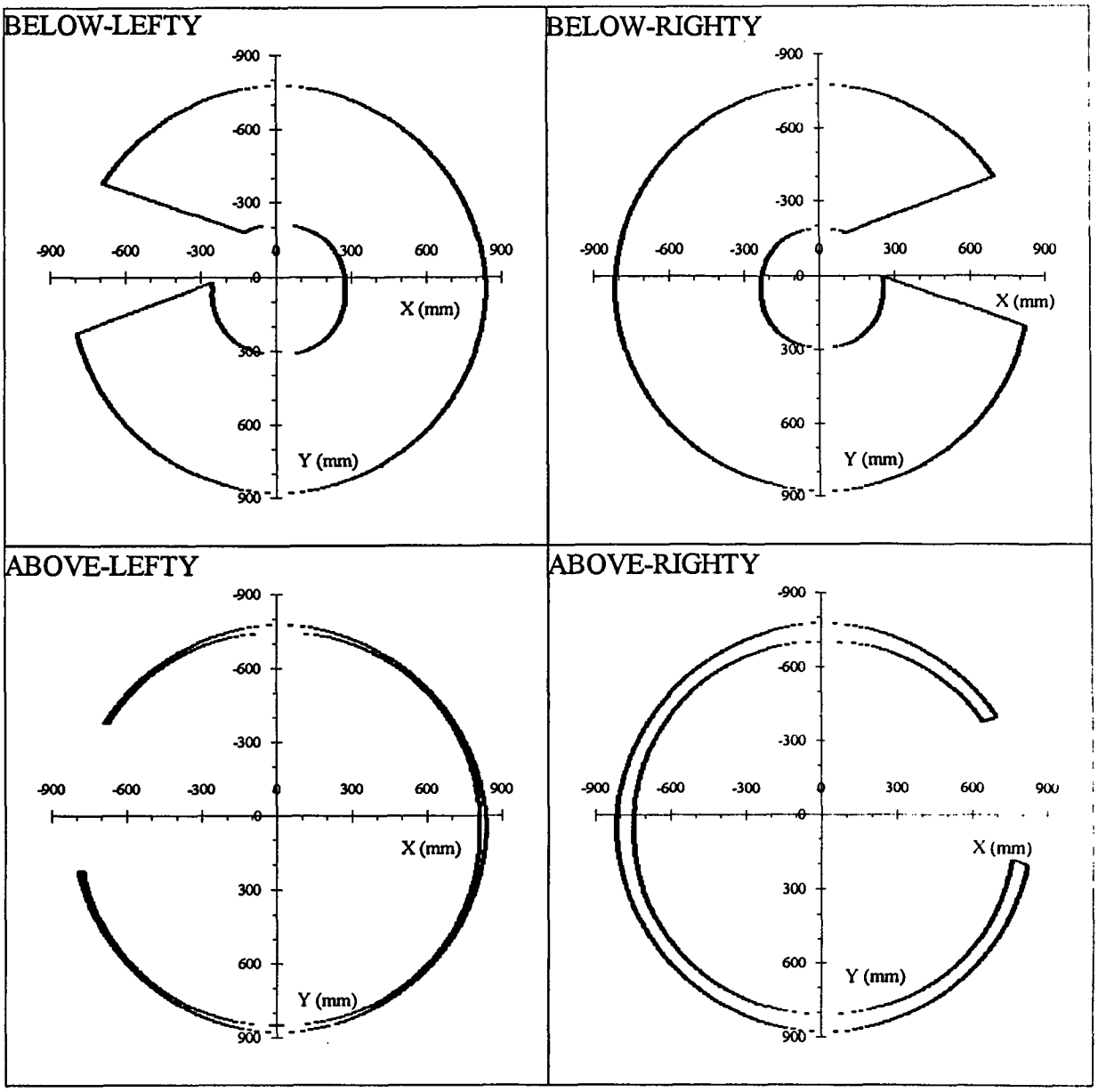


Figure 2.5 PUMA Robot Arm Work Envelope



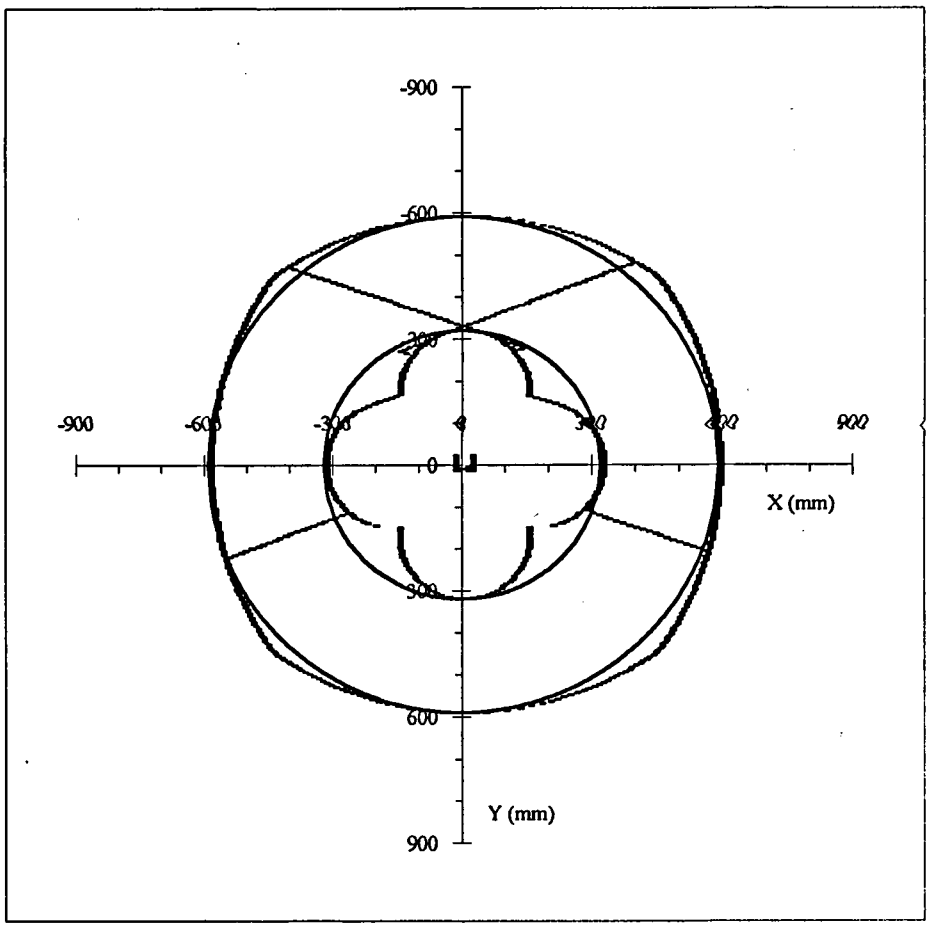
PUMA Work Envelopes in the X-Y plane where Z = 576

Figure 2.6 PUMA Work Envelopes With Respect to Cue Tool in Various Postures



PUMA Work Envelopes in the X-Y plane where Z = 273

Figure 2.7 PUMA Work Envelopes With Respect to the On-board Camera in Various Postures



Combined PUMA work envelope with respect to the cue in BELOW postures can be modelled as a disk where the inner circle has diameter 320 mm and the outer circle has diameter 590 mm.

Figure 2.8 Modelling of PUMA Work Envelope

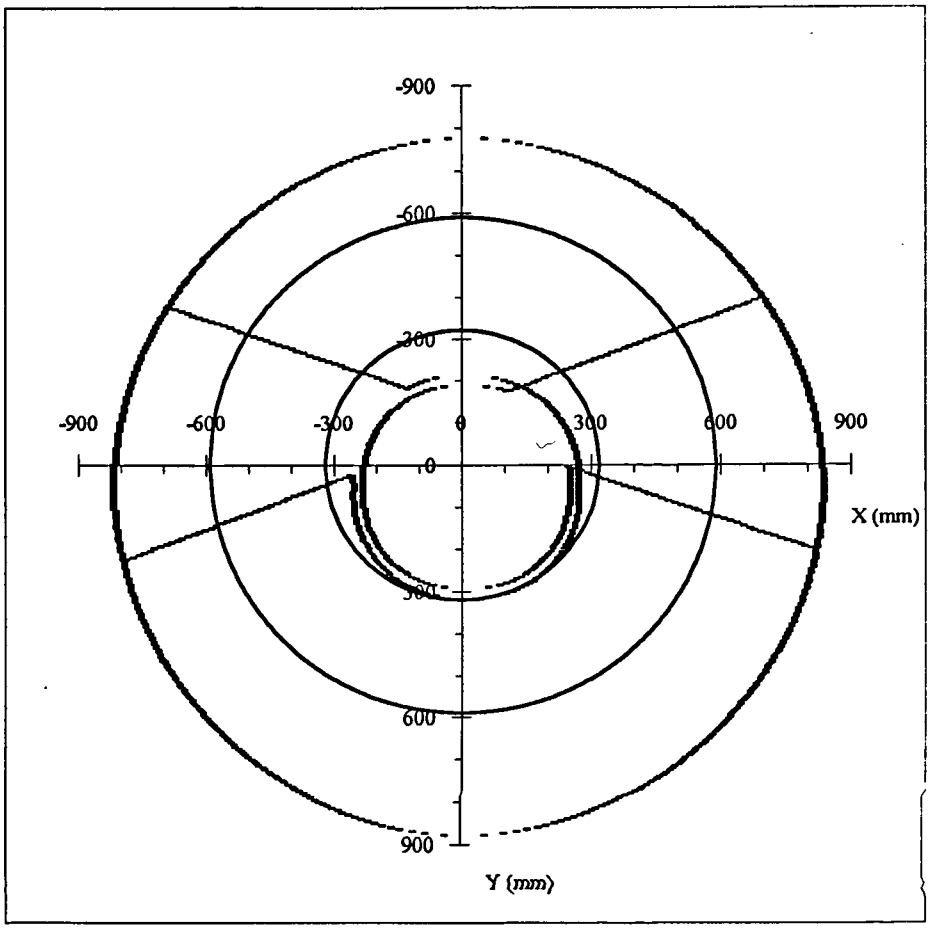
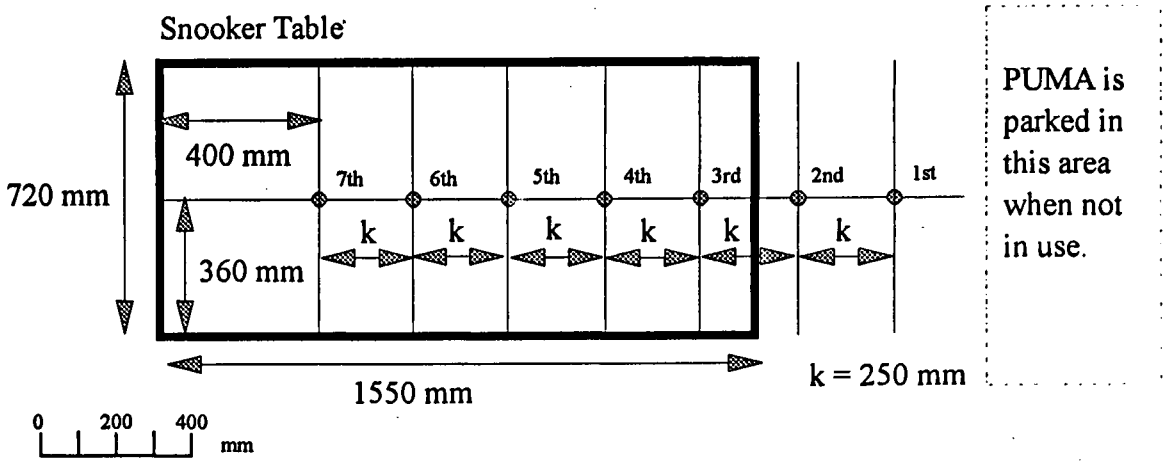
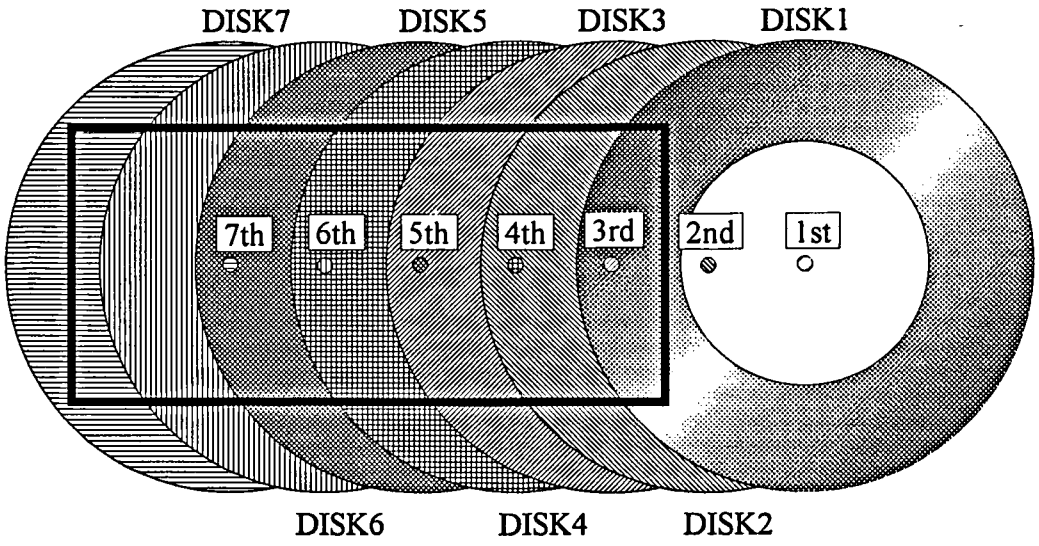


Figure 2.9 PUMA Work Envelope With Respect to On-board Camera in BELOW Postures and Model PUMA Envelope

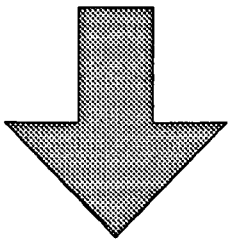


(a) Definition of SKF Base Points



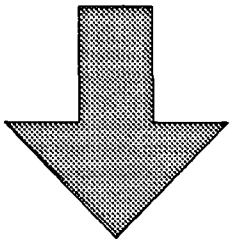
(b) Coverage of the 7 PUMA Work Envelopes

Figure 2.10 Extended PUMA Work Envelope Generated by Using 7 PUMA Base Points



Input :
(x, y) position of a snooker ball on the table in WORLD co-ordinates

```
FOR i = 1 to 7 do
    PUMA_X = X - XBase[i]
    PUMA_Y = Y - YBase[i]
    D = SQRT ( SQR ( PUMA_X ) + SQR ( PUMA_Y ) )
    IF D <= 590 mm
    THEN RETURN ( i, PUMA_X, PUMA_Y )
END
RETURN ( ERROR , point (x, y) is out of reach )
```

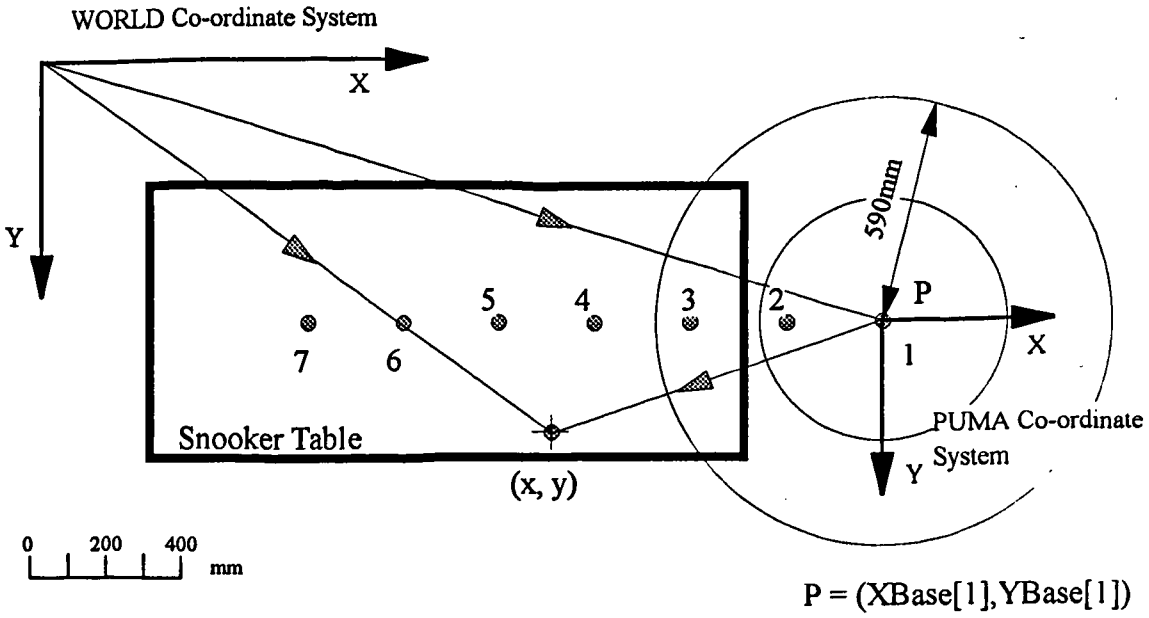


Output :
SKF base point number i, i = 1..7
(PUMA_X, PUMA_Y) target point in PUMA co-ordinate system

or

ERROR

Figure 2.11 Incremental Algorithm to Determine PUMA target point and SKF Base Point Number



(x, y) = position of a snooker ball in WORLD coordinates

(XBase[i], YBase[i]) = origin of PUMA co-ordinate system when robot arm is at SKF base point i

The vector triangle above shows clearly that the point (x, y) in WORLD co-ordinates is equivalent to the point (x-XBase[1], y-YBase[1]).

In general, a point (x, y) in WORLD coordinates can be transformed into PUMA co-ordinates at SKF position i as :

$$(x\text{-}XBase[i],\ y\text{-}YBase[i]) \qquad i = \{1,2,3,4,5,6,7\}$$

Notes :

(1) At SKF base point i, $1 <= i <= 7$, the PUMA co-ordinate system origin in WORLD co-ordinates is (XBase[i], YBase[i]). In this particular example, i = 1.

(2) (x, y) is the co-ordinates of a ball on the snooker table, computed by the vision system.

(3) The fact that the WORLD and PUMA co-ordinate systems have the same orientation allows the point (x, y) to be tranformed from WORLD to PUMA co-ordinates simply as the difference between two vectors, as the vector triangle above shows. If their orientations are different, rotational transformation will be involved.

(4) The position of the WORLD co-ordinate system in relation to the snooker table is not important. However, the 7 SKF positions must be defined as specified in Figure 2.10 (a) in relation to the snooker table to gaurantee 100% coverage of the table by the PUMA.

Figure 2.12 Conversion of Point (x,y) From WORLD to PUMA Co-ordinates

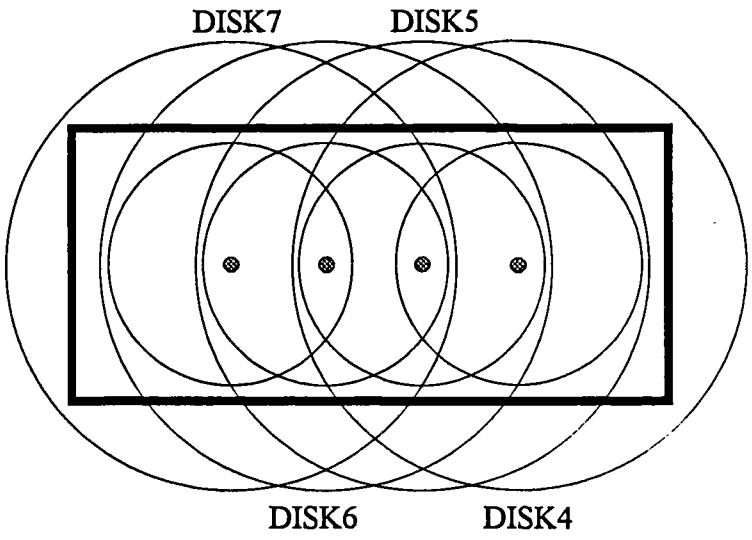


Figure 2.13 Combined PUMA Envelope With 4 SKF Base Points

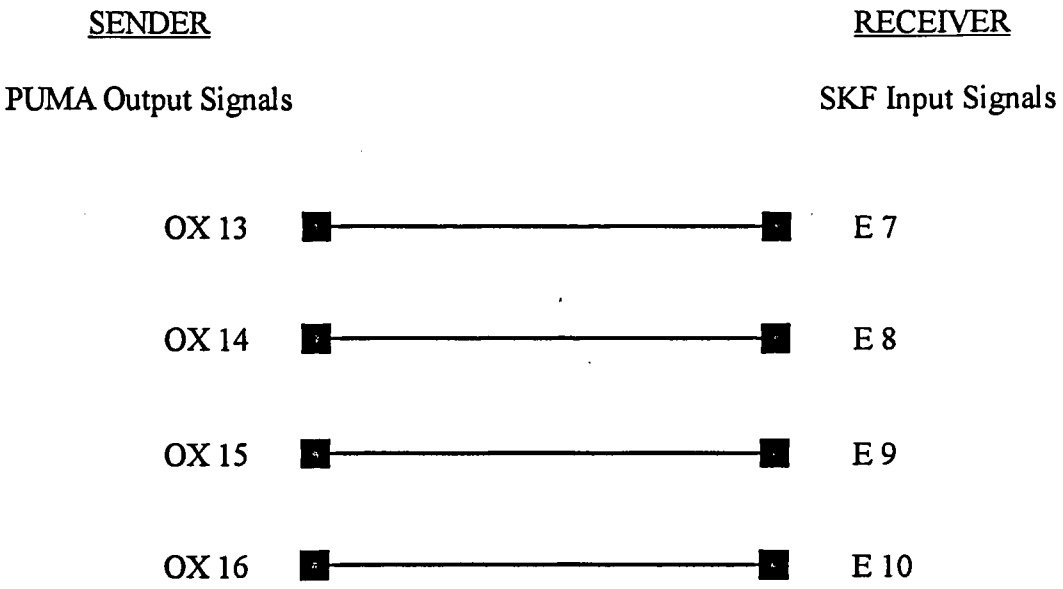
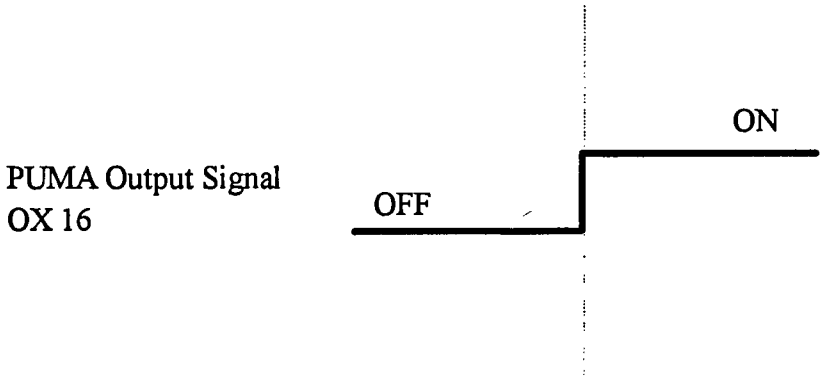


Figure 2.14 Communication Lines Between PUMA and SKF



Leading edge of pulse triggers SKF program to decode input signal E7, E8, and E9 and start SKF motion to move to the specified point.

Figure 2.15 Pulse Signal Generated by PUMA Program

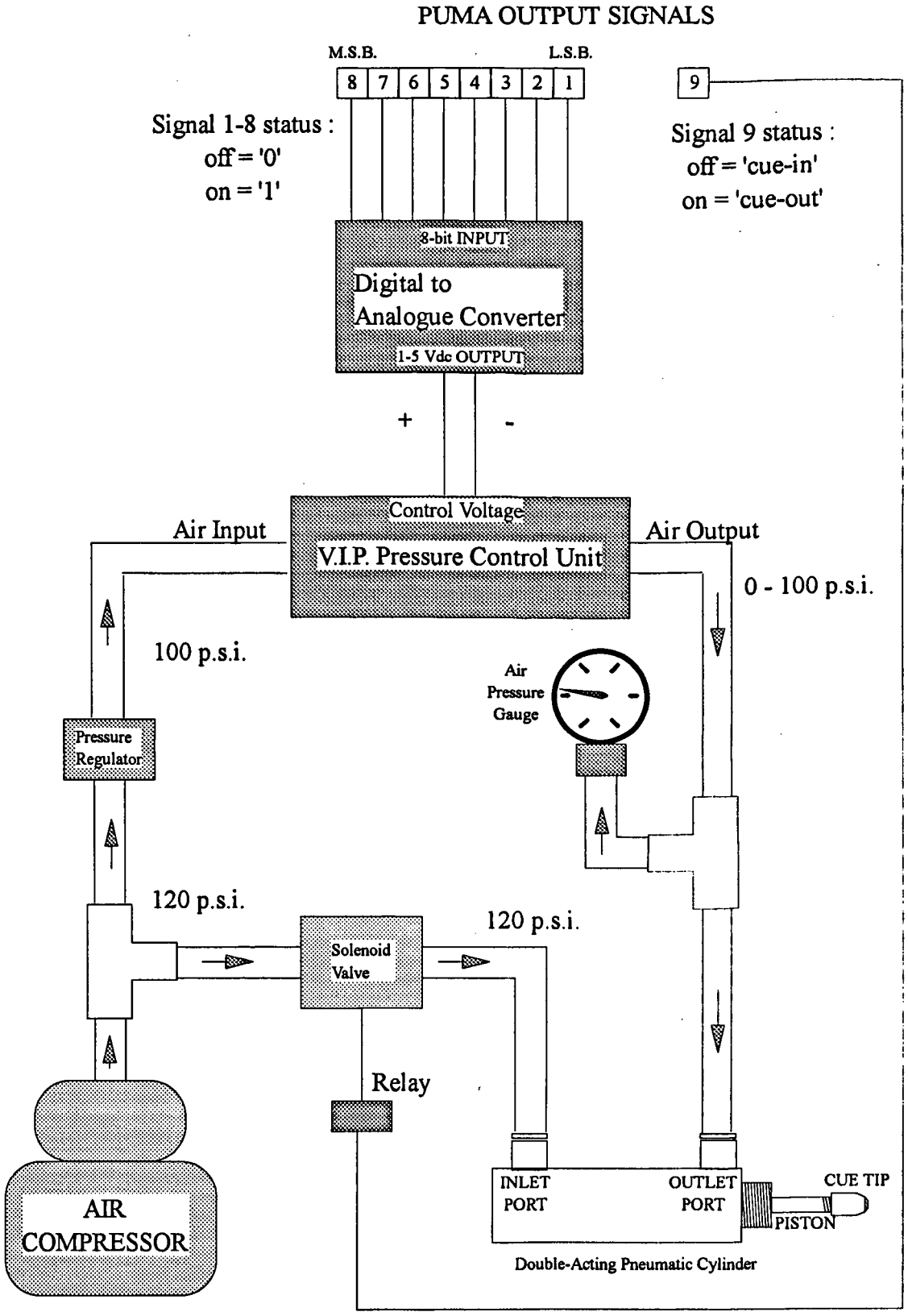
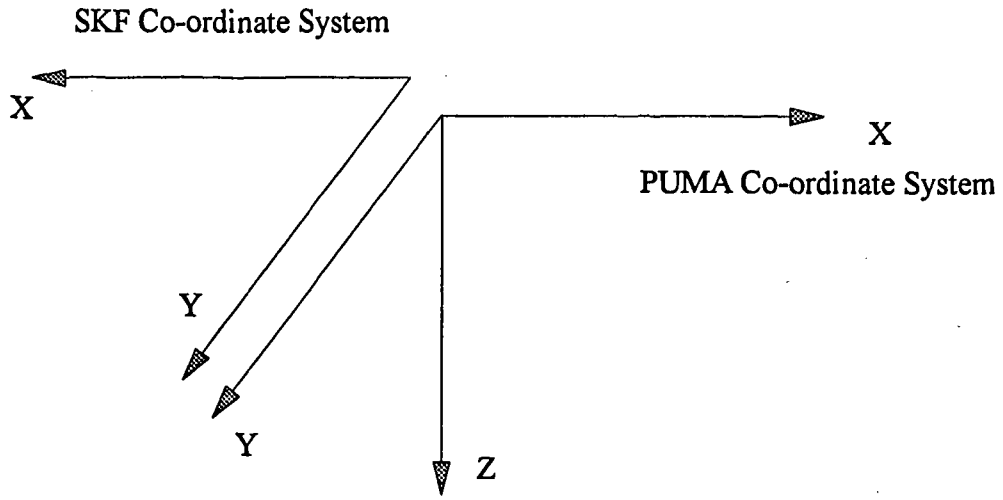


Figure 2.16 Pneumatics Control and Cue Actuation Mechanisms



Note : Origin of PUMA co-ordinate system is the intersection of joint axes 1 & 2.
Origin of SKF co-ordinate system is arbitrary before calibration, after which the SKF origin coincides with the WORLD origin.

Figure 2.17 SKF and PUMA Co-ordinate Systems

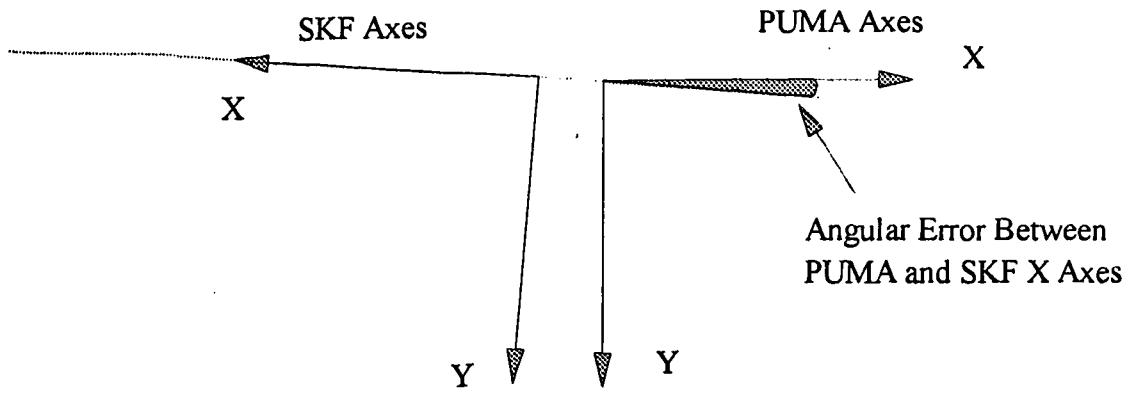


Figure 2.18 Plan View of PUMA and SKF Co-ordinate Systems

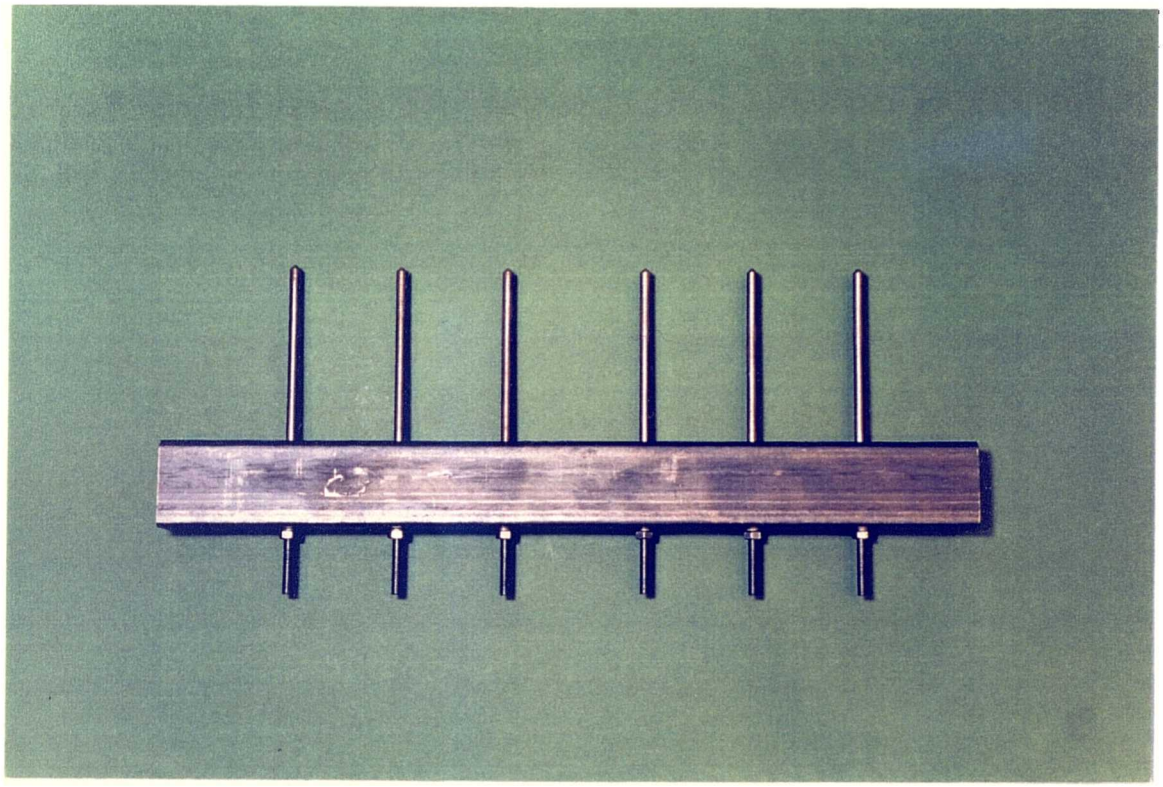


Figure 2.19 Axis Alignment Tool

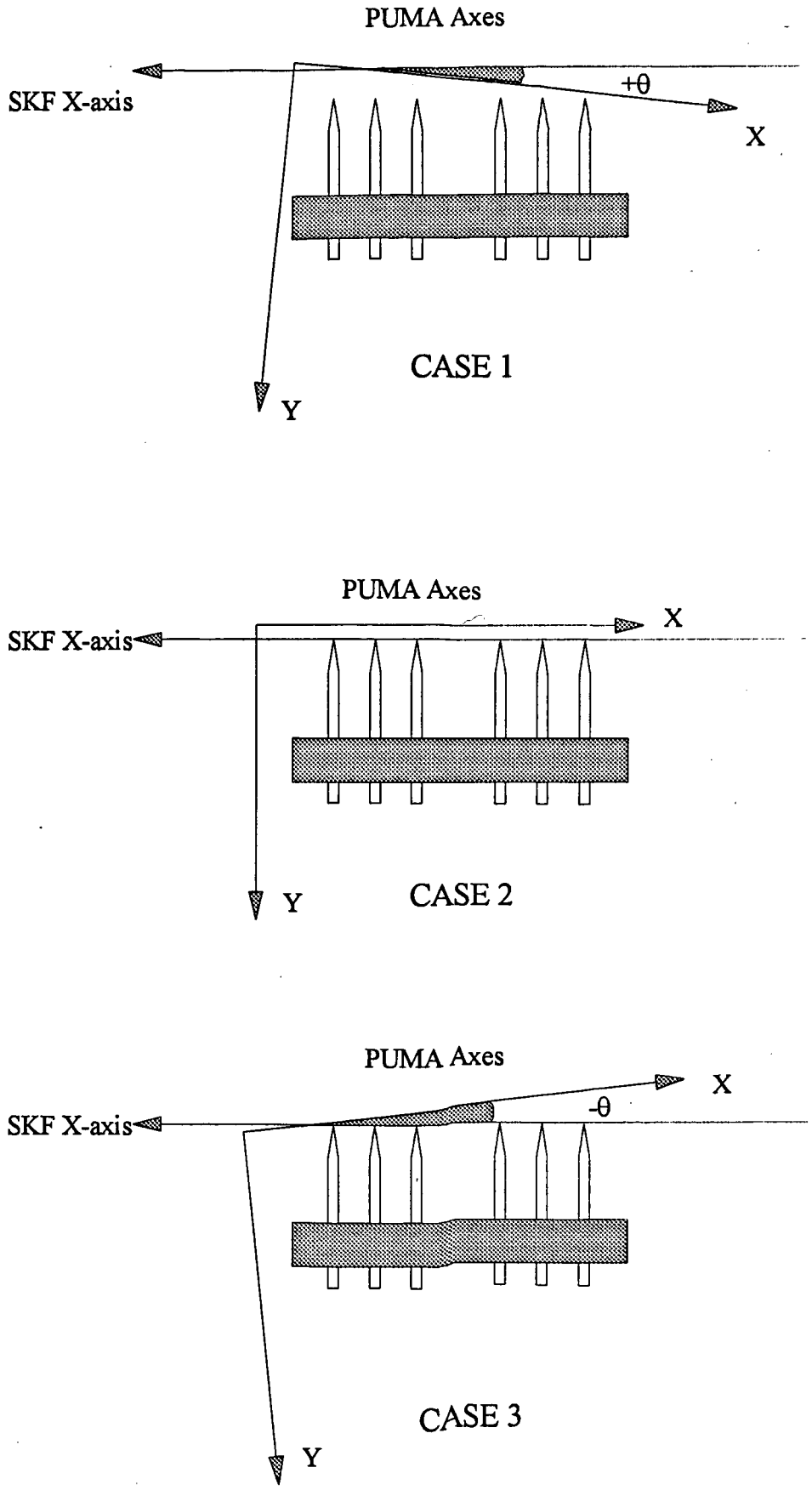
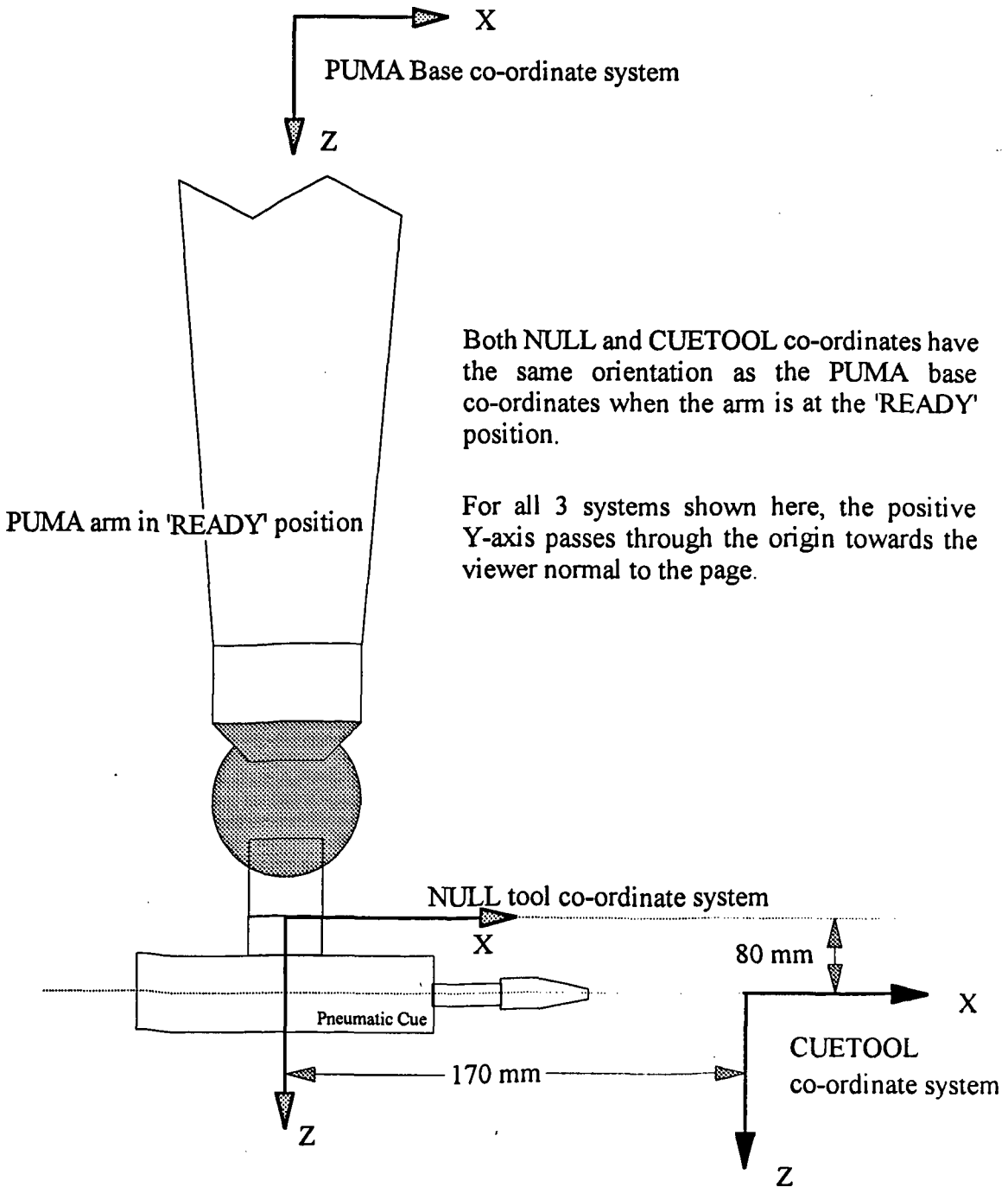


Figure 2.20 Plan Views of PUMA and SKF Axes Alignment



Tool Definitions : NULL = (0,0,0,90,-90,0)
CUETOOL = (170,0,80,90,-90,0)

Figure 2.21 PUMA Tool Definitions

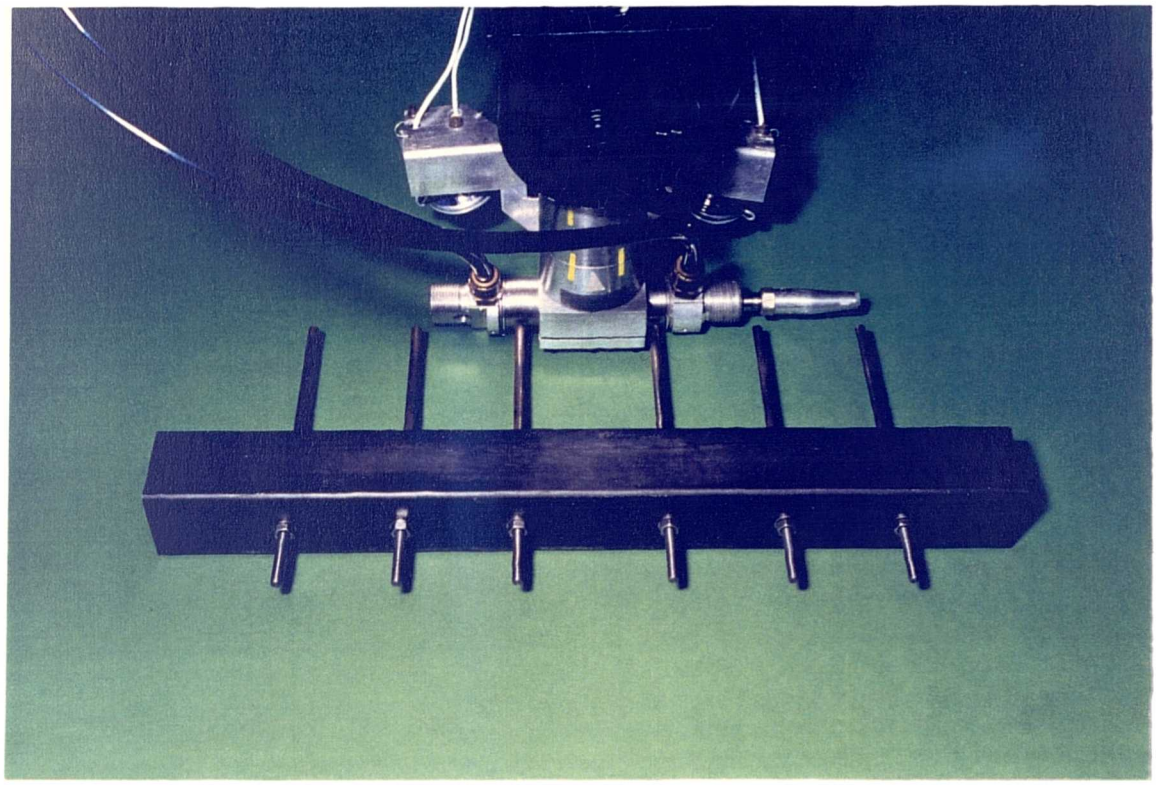


Figure 2.22 Aligning the Pneumatic Cue

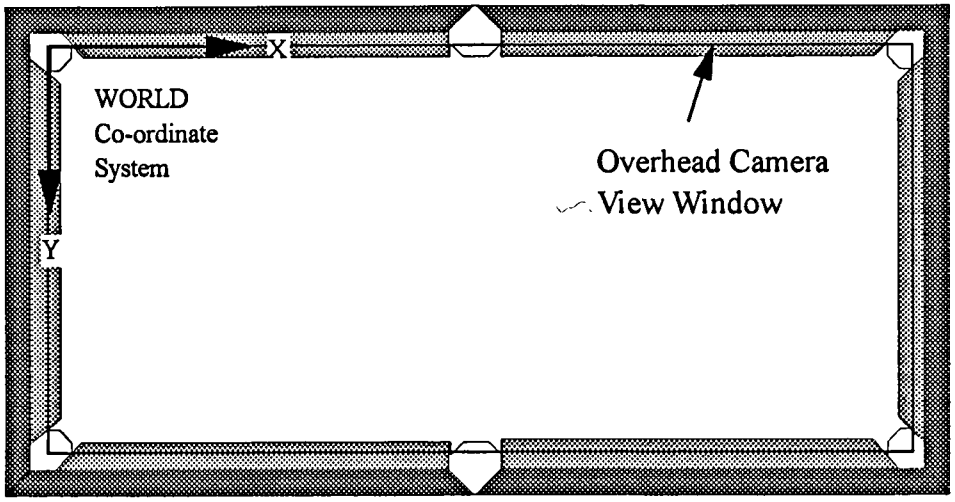
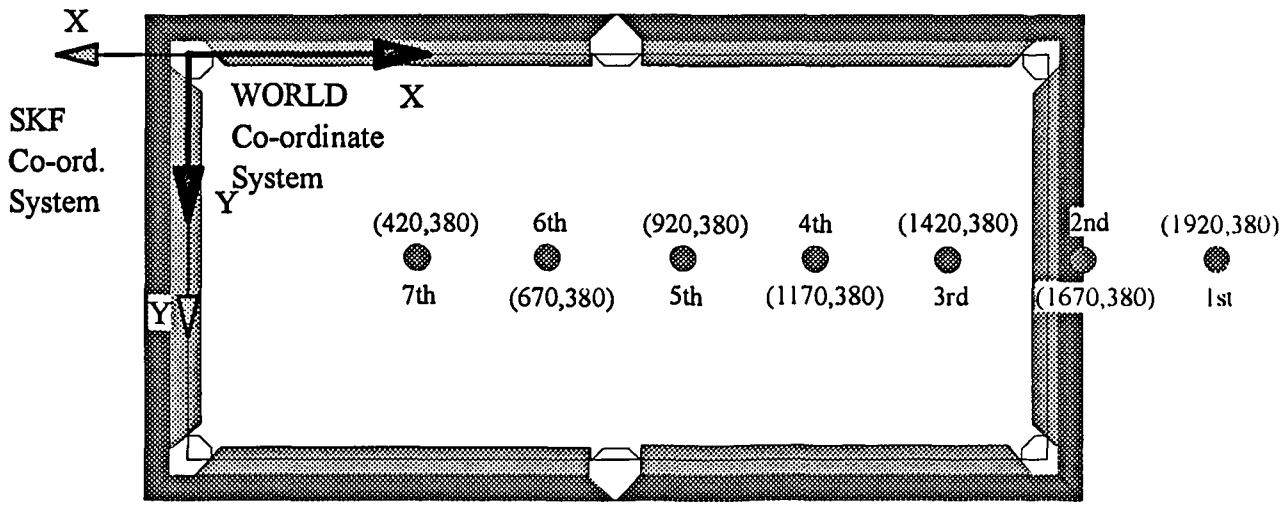


Figure 2.23 Definition of Overhead Camera View Window and WORLD Axes



● SKF Base Points over which the PUMA co-ordinate system origin can be positioned.

All (x, y) positions above are in WORLD co-ordinates (mm). Their equivalent co-ordinates in SKF system are listed below (computed using conversion factors determined in chapter 2.4.2) :

	Equivalent SKF Co-ordinates	
	X	Y
1st	-60927	12018
2nd	-52993	12018
3rd	-45060	12018
4th	-37127	12018
5th	-29194	12018
6th	-21261	12018
7th	-13328	12018

Figure 2.24 SKF Base Points in WORLD and SKF Co-ordinates

Chapter 3 Mathematical Analysis of Snooker

3.1 Expertise in Snooker

The true expertise in advance snooker play lies in the ability to position the cue ball at a favourable position after potting a ball rather than just the accuracy or success rate of a single shot. The ability to string together a series of shots is essential to the building of a ' *break* ', the continual potting of snooker balls whereby points are amassed. To build a good ' *break* ', the player has to control the positioning of the cue ball after a ball is potted. In other words, a skilled player actually manipulates the whereabouts of the cue ball after potting a ball by varying the cue force and where the cue tip strikes the cue ball. At an even more advanced level, the player is capable of using the cue ball to cannon into 1 or more balls to create a favourable game situation from an otherwise unfavourable one e.g. to move a ball away from the cushion.

Thus two cueing variables that have a direct bearing on positional play are the cue force and the striking point on the cue ball. The first variable has a relatively straightforward effect on the behaviour of the cue ball. The initial velocity of the cue ball is directly proportional to the impulsive force delivered by the cue. The amount of momentum retained by the cue ball after impact with the target ball depends on the angle of collision. In general the harder the cue ball is struck, the further it will travel after impact with the target ball. The effect of varying cue force in a plain shot is illustrated in figure 3.1. A plain shot is one where the cue ball is struck dead centre by the cue horizontally through the ball centre.

The effect of varying the striking point on the cue ball is more intriguing. In a plain shot, no initial rotation is exhibited by the cue ball. However, the striking point can be above or below the centre point, resulting in top and bottom spin respectively; or to the left or right, resulting in left and right hand side, as seen in figure 3.2. Furthermore, rotation about the 2 axes in figure 3.2 can be combined to varying degrees, producing 'top with left-hand side', 'top with right-hand side', 'bottom with left-hand side', and 'bottom with left-hand side'. Figure 3.3 shows 9 typical striking points on the cue ball. It has to be emphasised that there are in fact infinite combinations of top/bottom and left/right hand side and the resultant spinning is also infinitely variable depending on the cue force, the distance between the striking point and the ball centre, and also the distance between the cue ball and target ball. Note that if the cue ball is struck off-centre, it still travels in the direction of the cue, but its behaviour on impact with another ball or the cushion is altogether different from that of a plain shot. Figure 3.4 shows some typical effects of spins.

It has been decided from the outset that the robot snooker player system is going to play plain-shots alone for the following reasons :

(1) it is believed that the system should be able to play snooker at a reasonable level if the 1-step ahead game planning strategy implemented performs as intended.

(2) since there are infinite combinations of possible spin-effects, inclusion of spin-shots will lead to a combinatorial explosion of immense scale under the adopted game-planning strategy. The computing power required will also exceed that provided by the system of the current set-up.

(3) a simple pilot system incorporating plain-shots alone will better illustrate the feasibility and performance of the game planning strategy.

By limiting the number of decisions the system has to make to 2, i.e. the selection of shots and the cue force, the feasibility of transferring human skill into robotics is established. Modifications such as 2-steps ahead planning and inclusion of spin-shots can be introduced as additional system modules within the existing system. At the current pace of developments in robotics, artificial intelligence and computing, there is no doubt that applications of robotics in skilled operations can only become more widespread.

3.2 Dynamics of Snooker Ball Motion

In this section, the dynamic behaviour of a snooker ball on the snooker table is analysed. How a stationary snooker ball behaves when struck by a moving object will be studied. In a typical shot, two stationary balls are involved : the striking ball (which is always the cue ball in a game) and the target ball. The striking ball is first set in motion by the pneumatic cue towards the target ball. On impact, a certain amount of the striking ball's momentum is transferred to the target ball which then heads towards a pocket. The aim is to model the path of the cue ball as accurately as possible so that by varying the cue force, the position of the cue ball after the shot can be controlled. This modelling is broken down into several sub-problems. In this section, the motion of a snooker ball after acquiring an initial velocity on receiving an impulsive blow delivered by the pneumatic cue horizontally through the ball centre will be studied, assuming no obstacles lie in its path. Some of the work described below can also be found in a previously published paper **[Cheung90]**.

When the pneumatic cue strikes a stationary ball horizontally through its centre, the ball receives an impulsive force resulting in an instantaneous change in momentum, giving it an initial instantaneous velocity v_0. The cue ball initially slides on the table surface with no rolling. But this state of motion changes as this forward sliding motion of the ball is opposed by the frictional force between the ball and the green baize covering the snooker table. Figure 3.5 is a force diagram showing how friction acts against ball motion. Suppose F_S *is* the frictional force acting against the ball motion, it then follows that :

$$F_s = -\mu mg$$

where μ = coefficient of friction between snooker ball and table
m = mass of snooker ball
g = acceleration due to gravity

The direction of the ball's motion is treated as positive. Force F_S has two effects on the ball : apart from decelerating the moving ball in a linear manner, it also produces a moment about the centre of the ball which causes rolling of the ball to begin.

Using Newton's second law, the linear deceleration of the ball can be deduced as :

$$\dot{v} = F_s / m$$
$$\Rightarrow \dot{v} = -\mu mg / m$$
$$\Rightarrow \dot{v} = -\mu g \qquad \textbf{Eq. } (3.1)$$

On the other hand, the frictional force F_S produces a moment about the ball centre resulting in a torque τ which causes the ball to rotate in the direction of its travel. The torque τ exerted by F_S about the ball centre is :

$$\tau = |F_s| \times r$$

$$\Rightarrow \tau = \mu mgr \qquad \text{where } r \text{ is the ball radius}$$

Note that τ is treated as positive because the resultant angular acceleration is in the positive direction. The moment of inertia I of a snooker ball, which is a solid sphere, is defined as :

$$I = {}^{2mr^2}/_{5}$$

Knowing the torque τ and moment of inertia I, the angular acceleration $\dot{\omega}$ of the ball can be computed as :

$$\dot{\omega} = {}^{\tau}/_{I}$$

$$\Rightarrow \dot{\omega} = {}^{\mu mgr}/_{2mr^2/5}$$

$$\Rightarrow \dot{\omega} = {}^{5\mu g}/_{2r} \qquad \textbf{Eq. } \textit{(3.2)}$$

Immediately after the ball was struck, its linear velocity is v_0 with zero angular velocity. But given Eq. *(3.1)* and *(3.2)*, its linear and angular velocity after time t can be computed as :

$$v = v_0 + \dot{v}t = v_0 - \mu gt \qquad \textbf{Eq. } \textit{(3.3)}$$

$$\omega = 0 + \dot{\omega}t = {}^{5\mu gt}/_{2r} \qquad \textbf{Eq. (3.4)}$$

As the linear velocity v decreases and the angular velocity ω increases, there will come a point when the ball exhibits pure rolling motion, i.e. :

$$v = r\omega \qquad \textit{when } t = T_1 \qquad \textbf{Eq. } \textit{(3.5)}$$

This is a very significant point in the ball's motion which marks the end of the sliding phase and the beginning of the rolling phase. Once the sliding of the ball stops, the frictional force F_S ceases to affect the rolling ball.

Suppose the sliding phase ends at time T_1, the linear and angular velocity of the ball at T_1 are :

$$v_1 = v_0 - \mu g T_1 \qquad \textbf{Eq. } (3.6)$$

$$\omega_1 = {}^{5\mu g T_1}/_{2r} \qquad \textbf{Eq. } (3.7)$$

Substituting Eq. *(3.6)* and *(3.7)* into Eq. *(3.5)*, the following is obtained :

$$v_0 - \mu g T_1 = r\left({}^{5\mu g T_1}/_{2r}\right)$$

$$\Rightarrow v_0 = {}^{5}/_{2}\,\mu g T_1 + \mu g T_1$$

$$\Rightarrow T_1 = {}^{2v_0}/_{7\mu g} \qquad \textbf{Eq. } (3.8)$$

Substituting Eq. *(3.8)* back into Eq. *(3.6)* and *(3.7)*, v_1 can be computed as :

$$v_1 = {}^{5v_0}/_{7} \text{ and } \omega_1 = {}^{5v_0}/_{7r} \qquad \textbf{Eq. } (3.9)$$

It is most interesting that the sliding phase of the ball motion always ends when the ball has lost ${}^{2}/_{7}$ of its initial velocity irrespective of the coefficient of friction.

From time T_1 onwards, the ball exhibits pure rolling motion until the rolling resistance exerted onto the ball by the green baize finally stops it. Let $\dot{v}_r$ be the resultant linear deceleration acting against the ball's rolling motion. It is known that at the beginning of the rolling phase, the ball's velocity is ${}^{5v_0}/_{7}$. The end of the rolling phase corresponds to the termination of the ball's motion and so the final velocity is zero. Knowing the initial and final velocity of the ball during the rolling phase and the deceleration the ball is subjected to, the duration of the rolling phase and the displacement of the ball can be determined. The complete scenario of a snooker ball's linear motion on the snooker table is summarised in figure 3.6.

3.3 Co-linear Collision Between Two Snooker Balls

Having modelled the behaviour of a moving snooker ball on the snooker table, the next task is to investigate what happens when a moving ball strikes a stationary ball head-on, i.e. the target ball centre lies centrally in the path of the striking ball.

Before proceeding further, some important assumptions have to be made. Minute variations in snooker ball dimensions and masses are ignored and that all snooker balls are treated as perfect spheres of the same physical dimension and mass. Collision between two snooker balls is modelled as partially elastic such that kinetic energy is neither fully conserved nor totally lost. Finally, only collision between a moving ball and a stationary ball is studied. The current system does not model any secondary ball collisions i.e. any collision other than that between the striking cue ball and the stationary target ball.

Suppose the striking ball is moving towards a stationary target ball along the line joining their centres. Let the instantaneous velocity of the striking ball and target ball just before impact be $v_{11}\ ms^{-1}$ and $v_{21}\ ms^{-1}$ respectively. Note that v_{21} equals zero since the target ball was stationary before impact. After impact, the velocity of the striking and target balls are $v_{12}\ ms^{-1}$ and $v_{22}\ ms^{-1}$ respectively, as shown in figure 3.7. Since the collision is not perfectly elastic, the balls separation speed is bound to be less then their approach speed, and this ratio is defined as their coefficient of restitution, R, i.e. :

$$R = \text{Speed of Separation} \Big/ \text{Speed of Approach}$$

$$\Rightarrow R = (v_{22} - v_{12}) \Big/ v_{11} \qquad \textbf{Eq. } (3.10)$$

Also, by the law of momentum conservation :

$$mv_{11} = mv_{12} + mv_{22}$$

$$\Rightarrow v_{11} = v_{12} + v_{22} \qquad \textbf{Eq. } (3.11)$$

Solving Eq. *(3.10)* and *(3.11)* for v_{12} and v_{22} :

$$v_{12} = v_{11}(1 - R)/2 \qquad \text{Eq. } (3.12)$$

$$v_{22} = v_{11}(1 + R)/2 \qquad \text{Eq. } (3.13)$$

Eq. *(3.12)* and *(3.13)* show some interesting cases of a moving snooker ball striking a stationary one, depending on the value of R. If the snooker balls were perfectly elastic, i.e. $R = 1$, then the striking ball would stop after impact while the target ball will take off with the velocity of the striking ball i.e. they exchange their velocities during collision. At the other extreme, if collision were perfectly inelastic, i.e. $R = 0$, then the two balls would behave as stuck together at half the velocity of the striking ball (note the conservation of momentum but not kinetic energy in this case). In reality, the majority of the momentum is transferred from the striking ball to the target ball after impact.

3.4 Oblique Collision Between Two Snooker Balls

In an oblique collision, θ, the angle of attack is defined as the angle between the striking ball path *L1* and the line of impact *L2* which passes through the target ball centre and the striking ball centre at the moment of impact. Line *L3* is defined as perpendicular to *L2* through the striking ball centre at impact, as in figure 3.8. The case of co-linear collision is actually a special case of oblique collision where $\theta = 0$. After an oblique collision, the target ball will move along the line *L2*. However, the direction of travel of the striking ball after impact is the current focus of attention which is an important piece of information in ball control and positional play.

By resolving v_{11} into $v_{11}\cos\theta$, the component along *L2* and $v_{11}\sin\theta$, the component along *L3*, an oblique collision can be treated as a co-linear collision where the velocity of the striking ball before impact equals $v_{11}\cos\theta$, see figure 3.9. Since the velocity component perpendicular to *L2* is unaffected by the impact, the velocities of the striking and target balls after impact along *L2* can be determined using Eq. *(3.12)* and *(3.13)*. To get the complete picture, it must be remembered that the striking ball has that extra velocity component along *L3*, the resultant striking ball velocity v_{12} is shown in figure 3.10.

Thus all oblique collisions can be treated as linear collisions where the striking ball has a velocity component perpendicular to the line of impact that remains constant immediately before and after impact. The magnitude of this component depends on the angle of attack θ. The direction of travel of the striking ball after impact is expressed as an angle α, relative to the line $L3$. The value of α can be calculated from the constituent vectors of v_{12} as $\tan^{-1}\left[{}^{(1-R)}/_{2\tan\theta}\right]$ which is independent of the striking ball's approach velocity.

After impact, motion of the target ball will be subjected to the same forces as shown in figure 3.5 and figure 3.6 where its velocity and direction of travel are shown in figure 3.10. Note that for a successful shot, the target ball has to receive sufficient momentum from the striking ball to reach the pocket which in turns defines the minimum cue force required for that particular shot. On the other hand, the striking ball will be assumed to be exhibiting rolling motion after impact and therefore not subjected to any frictional force. This assumption is based on the fact that only in extreme situations will the striking ball be still within the sliding phase of motion (see figure 3.6) at impact with the target ball. For example, when a ball is subjected to the maximum cue force delivered by the pneumatic cue, the magnitude of the sliding phase is no more than 300 mm. This means that in any impact, the distance between the striking ball and target ball must not exceed 300 mm if the striking ball is to be within the sliding phase of motion at impact. At any other cue force, this maximum separation is correspondingly reduced. Motion of a rolling ball is subjected to retardation in the form of rolling resistance alone, as in the rolling phase of ball motion in figure 3.6.

3.5 Effect of Snooker Table Cushion on Ball Motion

After the striking ball has collided with the target ball, they will travel in the directions shown in figure 3.10. Under normal circumstances, the target ball will be heading towards one of the six pockets on the table. The striking ball path, however, will possibly encounter one or more table cushions before coming to rest. Therefore the effect of the cushion on ball motion must be analysed in order to complete the modelling of the path of the striking ball.

By observation, a snooker ball hitting a cushion is quite similar to a light beam reflecting off a mirror where the incident and reflected angles are equal. On closer inspection, it is apparent that the cushion absorbs some of the momentum of the ball. The case when a moving snooker ball hits the cushion at right angle will be considered first. Like the collision between snooker balls, collision between a snooker ball and the cushion is partially elastic. Therefore R_c , the coefficient of restitution of impact between snooker ball and cushion, can be defined as :

$$R_c = \text{Speed of Separation} \Big/ \text{Speed of Approach}$$

$$\Rightarrow R_c = |v_{out}| \Big/ |v_{in}|$$

$$\Rightarrow |v_{out}| = R_c |v_{in}| \qquad \textbf{Eq. } (3.14)$$

v_{in} and v_{out} are the velocities of the ball immediately before and after impact with the cushion. Note that v_{in} and v_{out} are implicitly opposite to each other.

When a moving ball hits the cushion at an angle, part of the velocity component normal to the cushion is lost during impact with the cushion while the component parallel to the cushion stays unaltered. Positioning of the snooker table in the workcell dictates that a cushion must be parallel to either the X-axis or the Y-axis of the WORLD co-ordinates system. Thus for any ball hitting a cushion with velocity v_{in}, its velocity components in the X and Y direction will be computed. Depending on which cushion the ball hits, the velocity component normal to the cushion will be subjected to Eq. *(3.14)* and reversed. The computation of the rebound velocity of a ball hitting a cushion is summed up in figure 3.11.

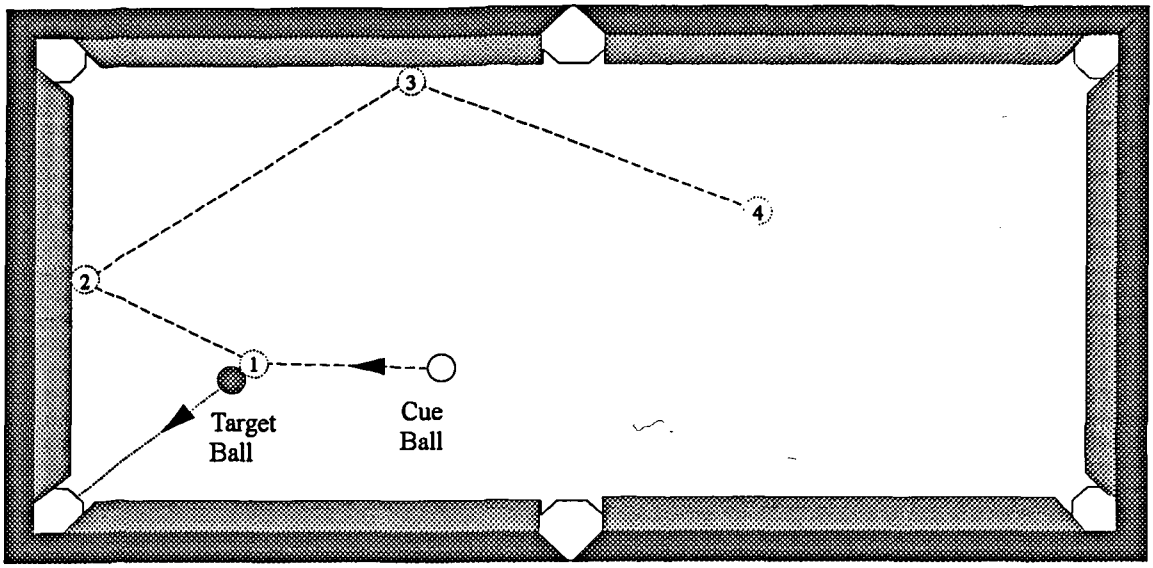
In chapter 3.2, it was stated that the motion of a ball subjected to an impulsive blow through its centre horizontally will first be opposed by friction and then rolling resistance when the sliding of the ball has stopped. Motion of a ball after rebounding off a cushion, however, is not subjected to retardation due to frictional force between the ball and the green baize. This is because the cushion is designed to make contact with the ball through its centre of percussion rather than its geometric centre, see figure 3.12. Any homogeneous sphere will exhibit immediate rolling motion on receiving an impulse through its centre of percussion. As a result, a snooker ball always rebounds off a cushion in a smooth rolling motion with negligible frictional loss.

Finally, an iterative algorithm is implemented to compute the final resting position of the striking ball after impact with the target ball. Given the position of the striking ball at impact and its velocity after impact, the snooker table is treated as an unbound area and the stopping position of the striking ball can be computed (remembering that after impact, the striking ball motion is assumed to be opposed by rolling resistance alone). The striking ball position at impact will be called P and the final position P_1. Since the positions of the four table cushion are pre-defined constants, whether the line segment PP_1 cuts any cushion can be checked. If a cut is detected, P_2, the position of the striking ball at impact with the cushion will be computed. Given P and P_2, the ball's velocity just before impact with the cushion and hence that after impact can be computed, as in figure 3.11. Now a new initial striking ball position and velocity are computed and the process to determine intersection between the striking ball path and any cushion is repeated. A graphical representation of this algorithm is shown in figure 3.13.

3.6 Conclusions

A model of the dynamic behaviour of the striking ball and target ball in a typical shot has been established. The motion of the striking ball on receiving an impulsive blow from the pneumatic cue, what happens when the striking ball collides with the target ball, and how the striking ball behaves on hitting a cushion are also analysed. The combination of these mathematical models enables the initial striking ball velocity to be related to its final position after a shot and forms the basis of ball control and game planning. An example of striking ball path modelling is presented in figure 3.14.

The determination of shot angle and shot difficulty will be covered in chapter 5 and how the modelling of the path of the striking ball after impact can be utilised in the one-step ahead game planning algorithm will be discussed in chapter 6. Also, four parameters that affects ball motion, namely the coefficient of friction between a ball and the snooker table surface, the linear deceleration as a result of rolling resistance, the coefficient of restitution of ball-to-ball impact, and the coefficient of restitution of ball-to-cushion impact are identified. The numerical values of these parameters, which directly affects the accuracy of the model, are not determined at this stage. In chapter 7, their values will be determined using genetic algorithms.



Target Ball Path Cue Ball Path

① Cue ball position during impact with Target Ball

② ③ ④ Approximate cue ball resting position after the shot is played with low, medium and high power.

Figure 3.1 Cue Ball Control by Varying Cue Force

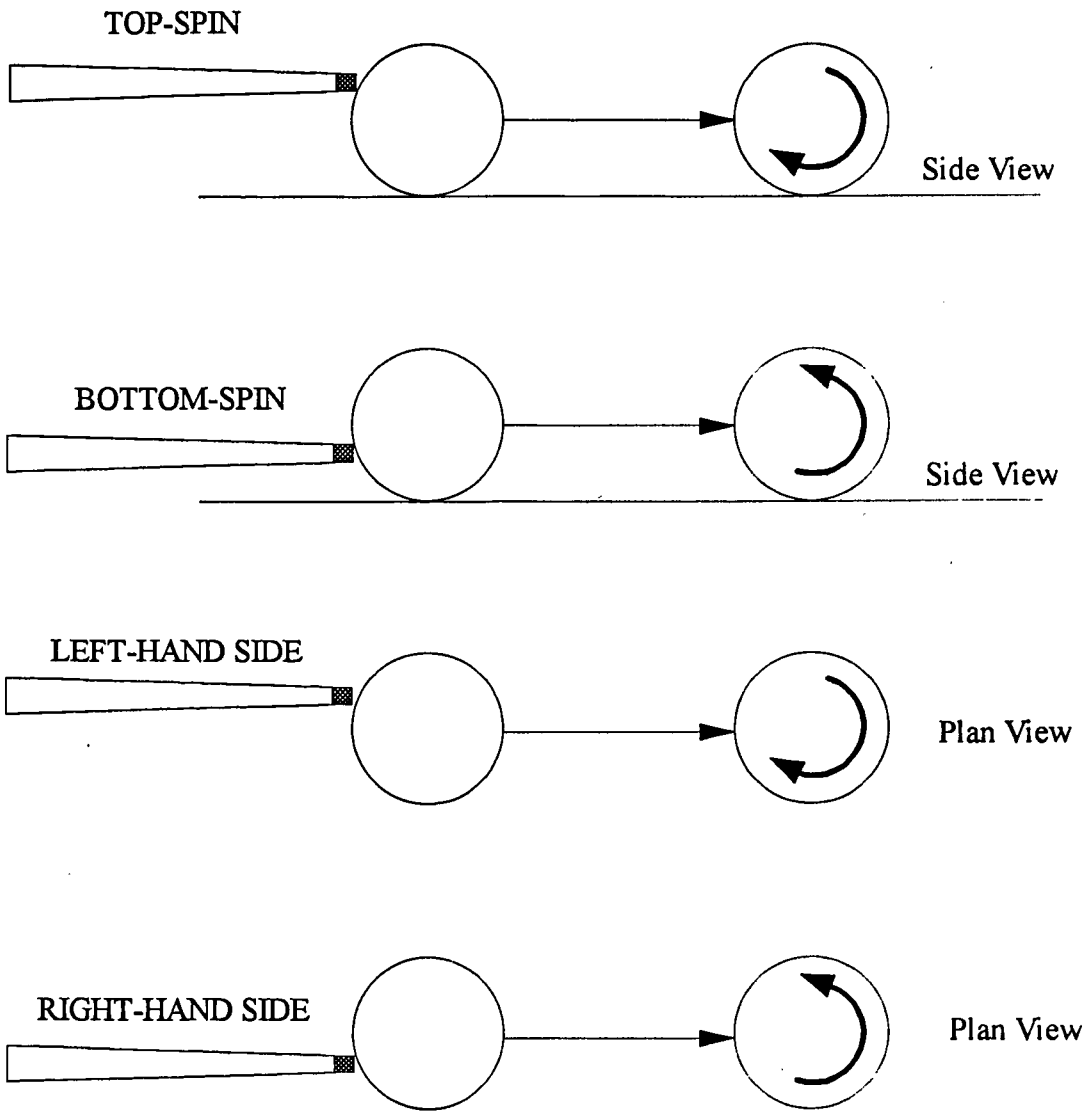
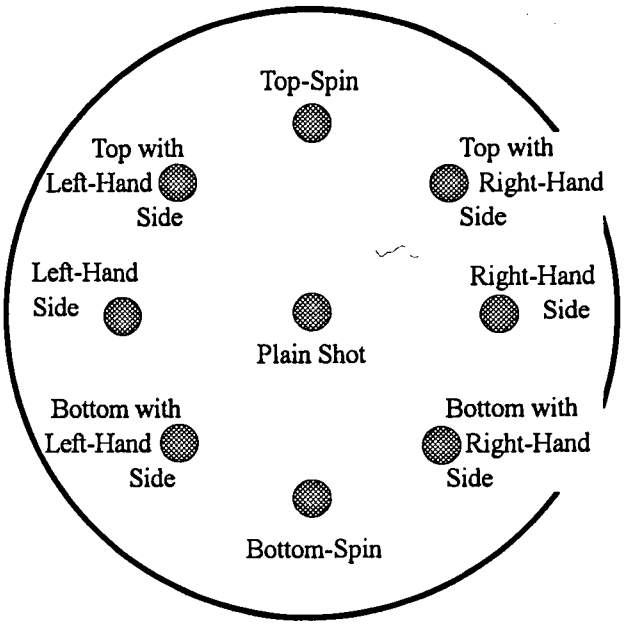
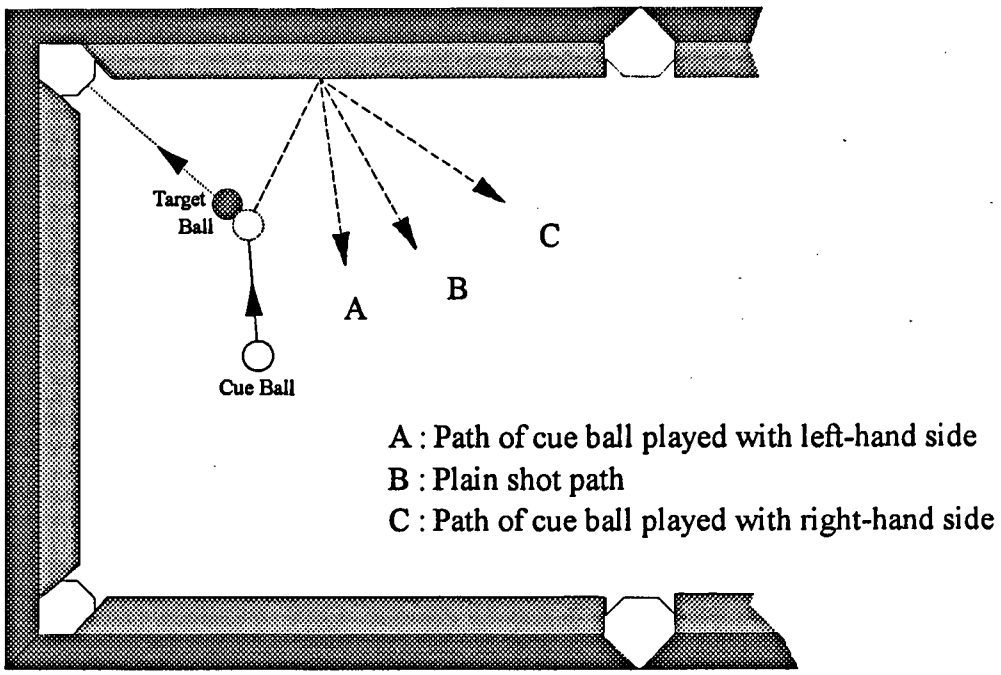


Figure 3.2 Four Basic Spin Effects



Cue ball as viewed along snooker cue

Figure 3.3 Relations Between Striking Points and Resultant Spin Effects



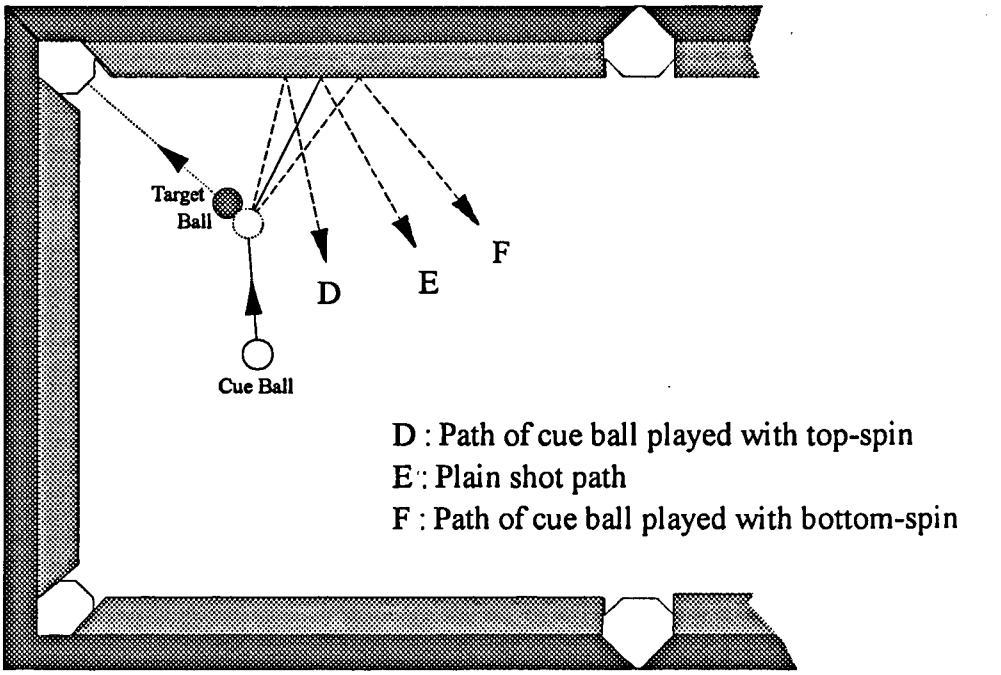


Figure 3.4 Examples of Cue Ball Path Control by Applying Spin

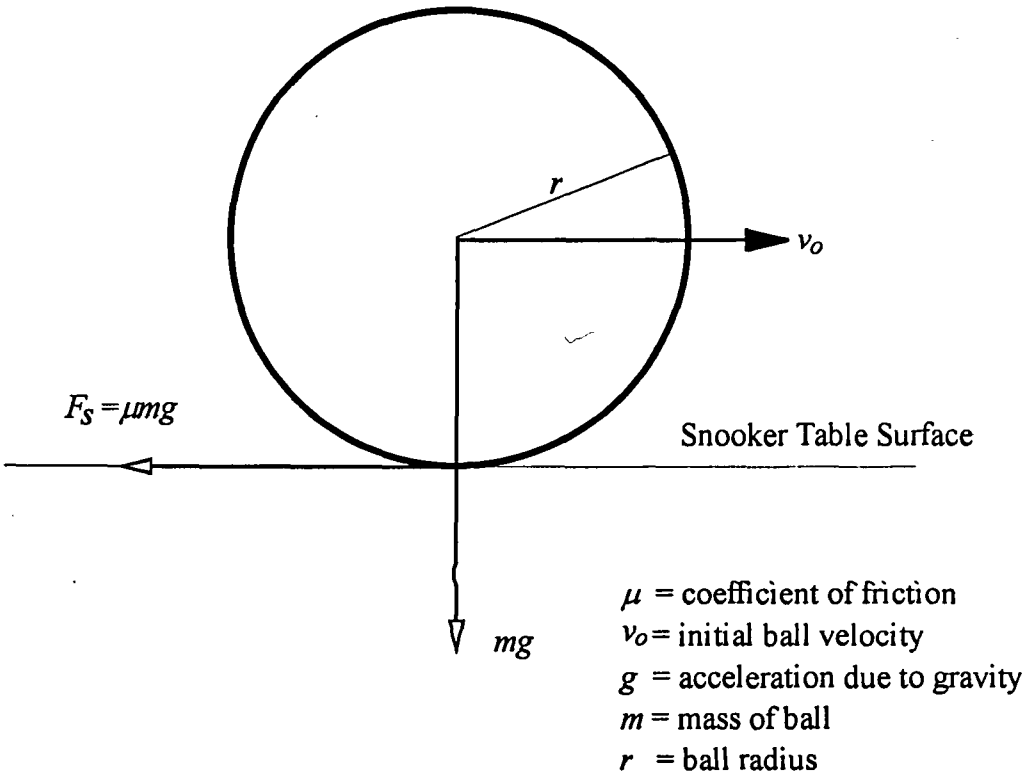
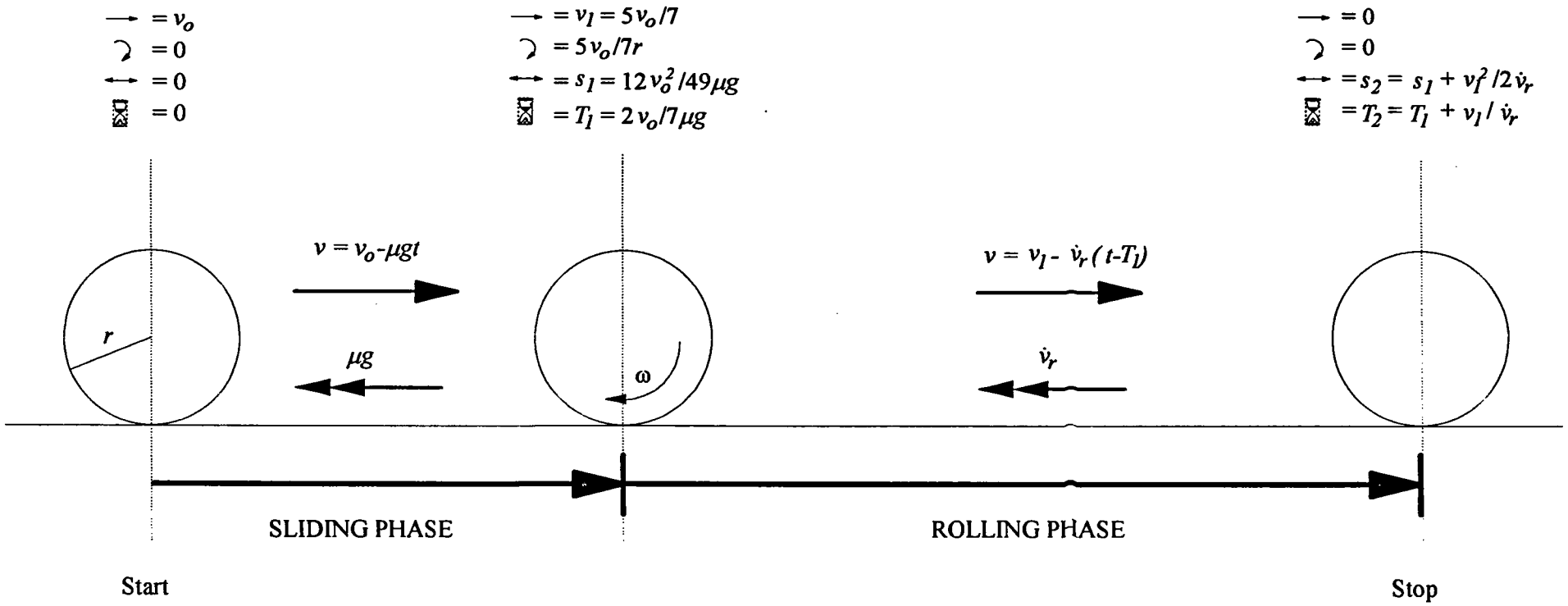


Figure 3.5 Forces Acting on a Sliding Snooker Ball



Legion : → velocity
↻ angular velocity
↔ displacement from starting position
⧖ elapsed time from start
← deceleration

v_o = initial velocity
r = ball radius
μ = coefficient of friction
g = acceleration due to gravity
$\dot{v}_r$ = resultant linear deceleration due to rolling resistance

Figure 3.6 Summary of Ball Motion on Snooker Table

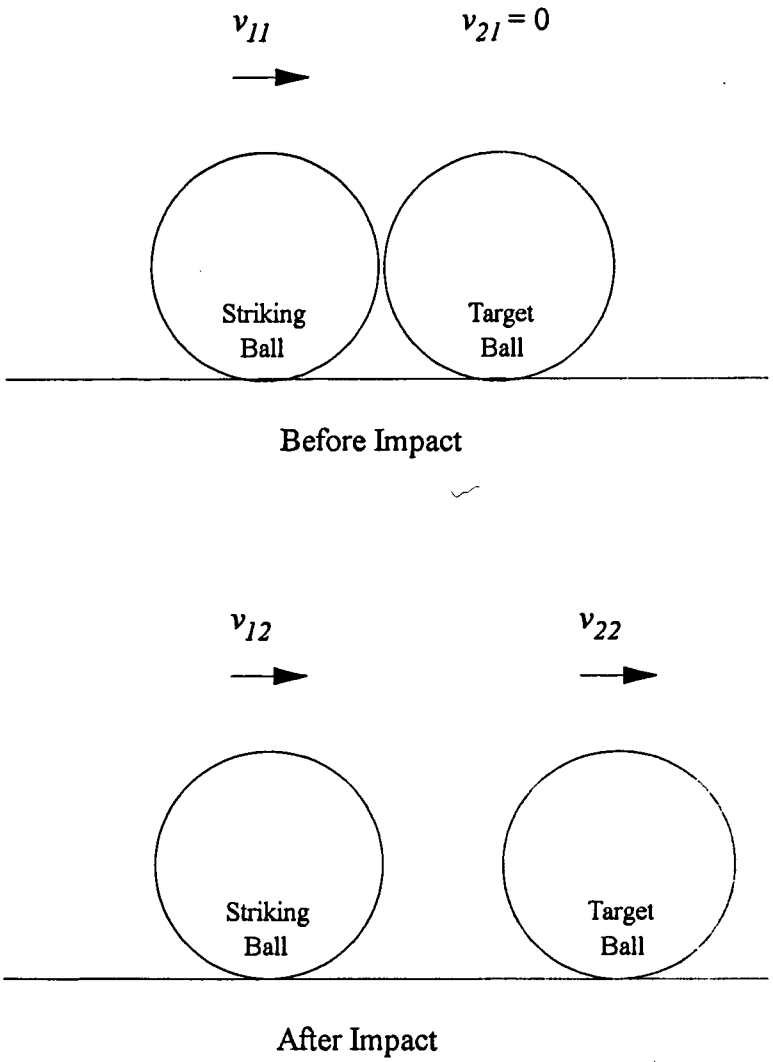


Figure 3.7 Velocities of the Striking and Target Balls in a Linear Collision

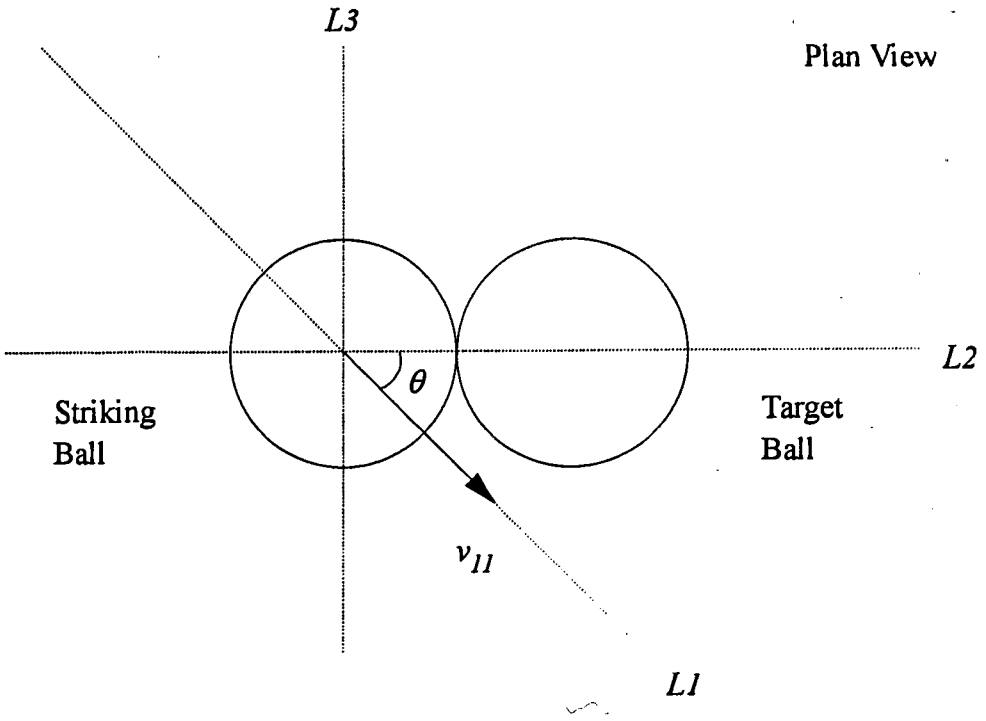


Figure 3.8 Oblique Collision Between a Moving and a Stationary Snooker Balls

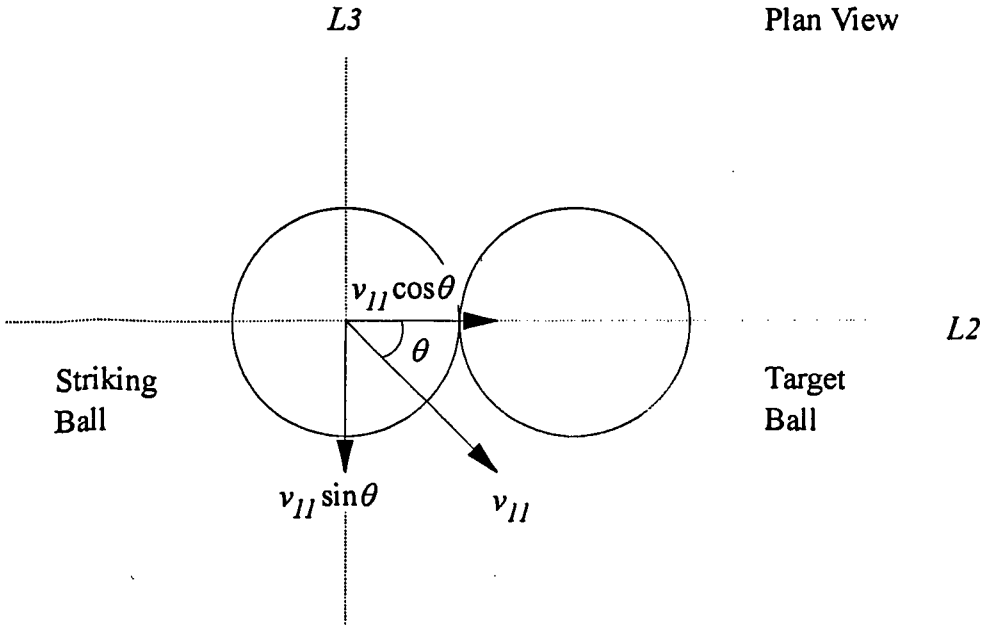


Figure 3.9 Velocity Components of Striking Ball Before Impact

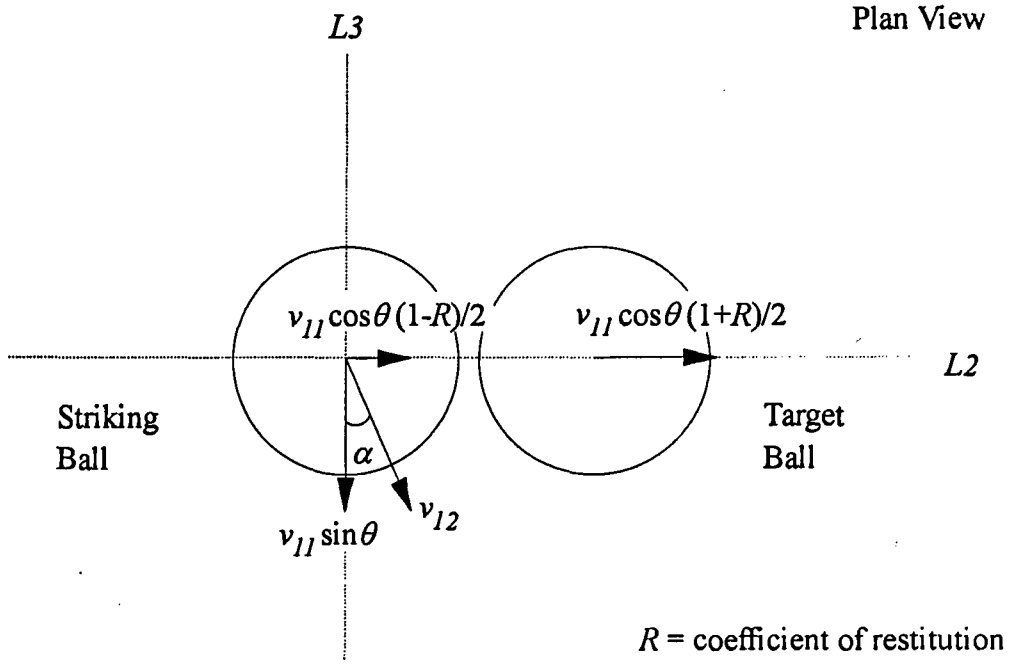
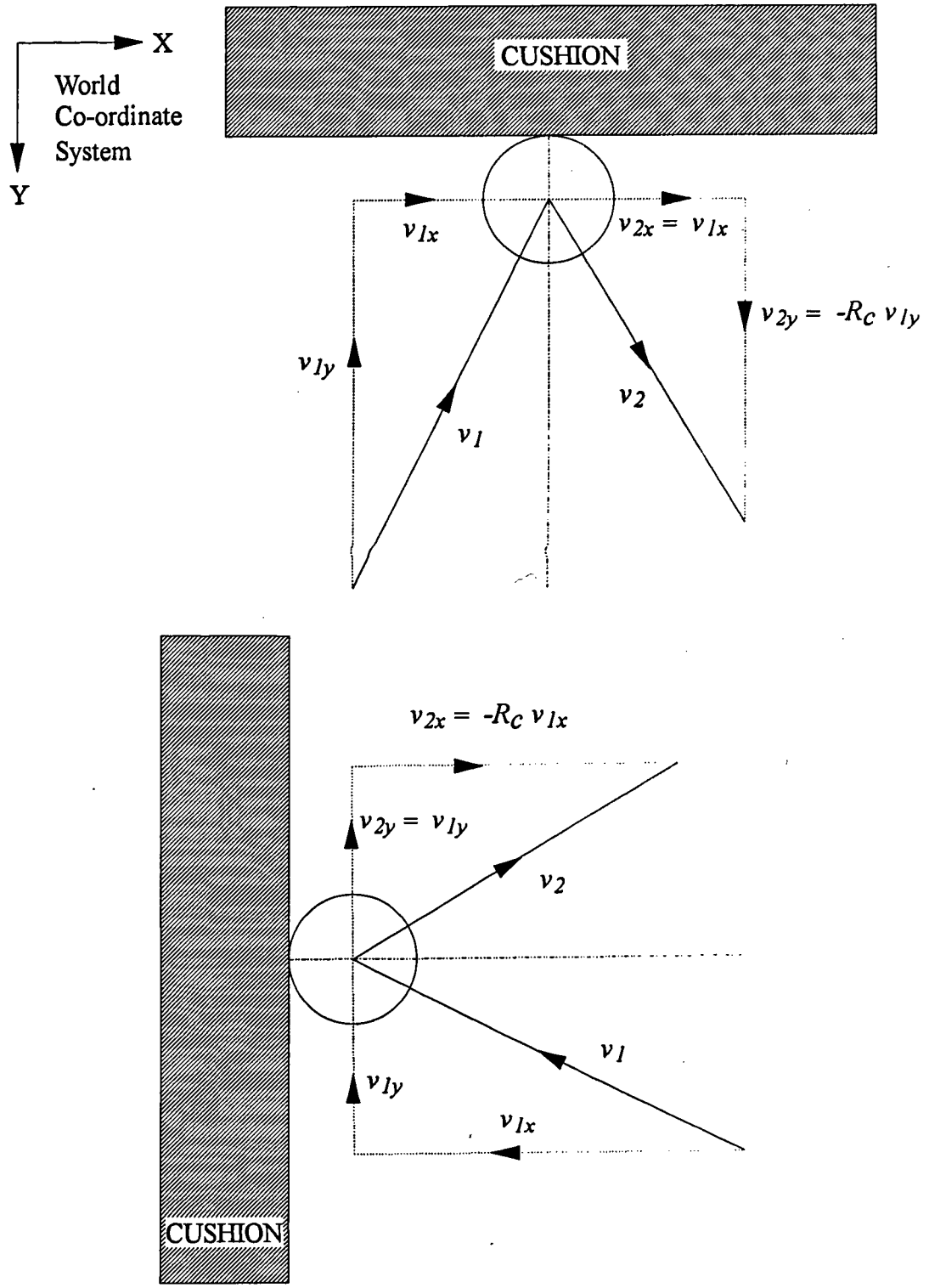
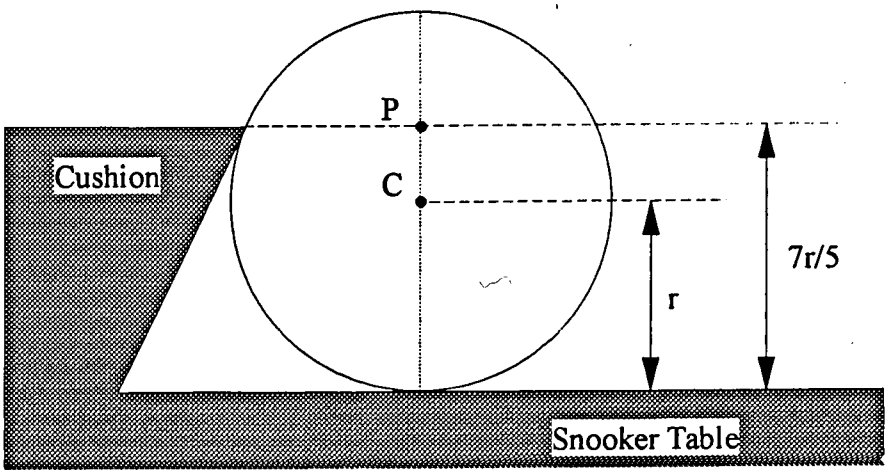


Figure 3.10 Velocities of Striking and Target Balls After Impact



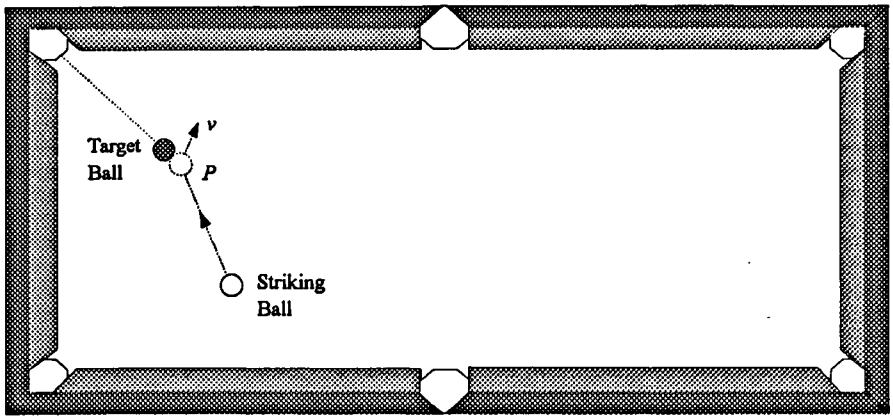
R_C = Coefficient of Restitution of Ball-Cushion Impact

Figure 3.11 Reflection of a Moving Snooker Ball Off a Cushion

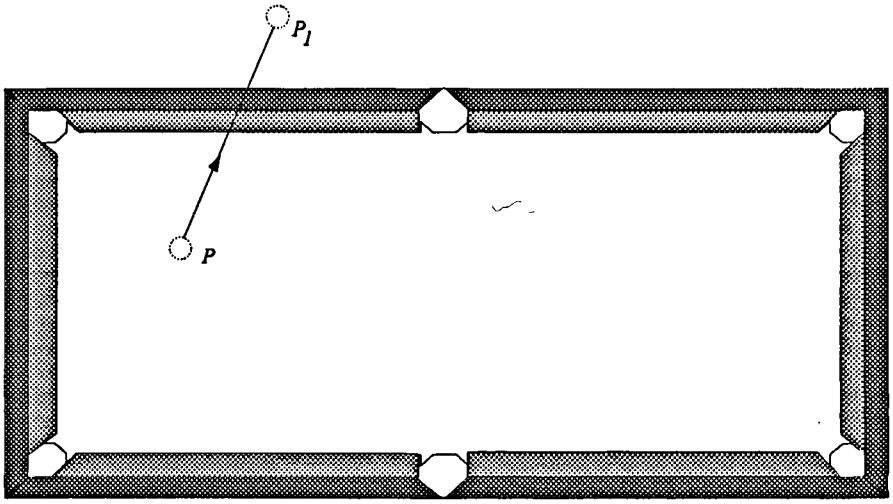


r = Ball Radius C = Ball Centre P = Centre of Percussion

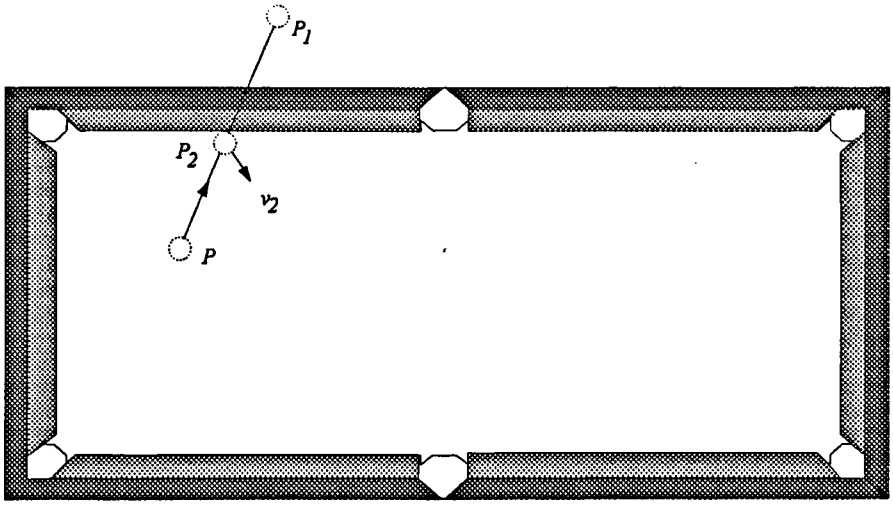
Figure 3.12 Contact Point Between Cushion and Ball



Step (i) Set P = striking ball position at impact
v = striking ball velocity after impact

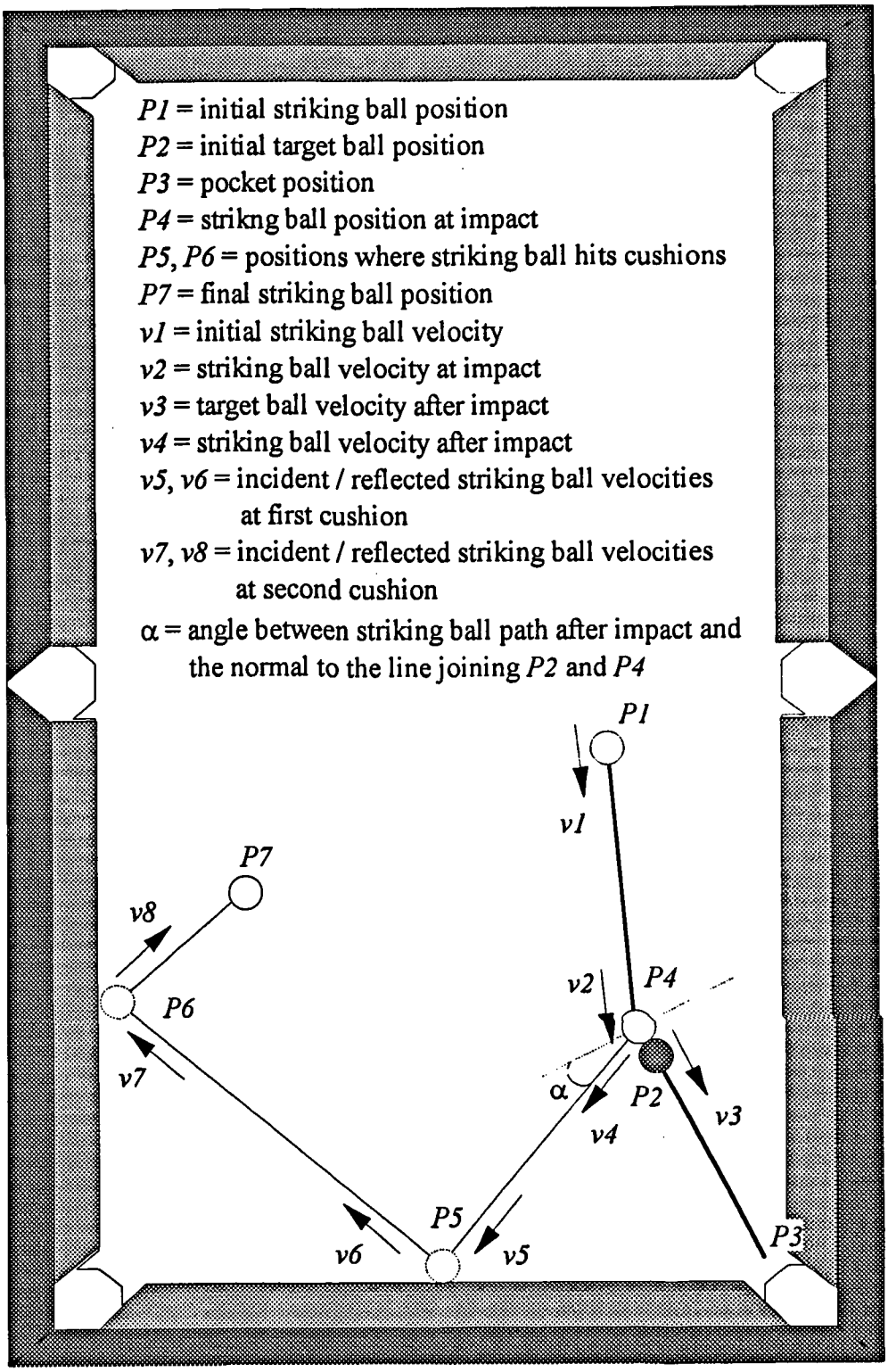


Step (ii) Compute final position of the striking ball after impact as if the snooker table is an unbound area.



Step (iii) IF PP_1 does not intersect any cushion THEN P_1 is the final position
ELSE compute P_2 and v_2
set $P = P_2$, $v = v_2$
GOTO Step (ii)

Figure 3.13 Iterative Algorithm for Finding Final Position of the Striking Ball After Impact



Note : Pocket and cushion positions are predefined constants.

Figure 3.14 Projection of Striking Ball Path in a Typical Shot

Chapter 4 Image Processing

The vision module within the system can be viewed as a black box which converts a digitised image of the snooker table with balls of various colours on it into a list of ball co-ordinates and their corresponding colours. In practice, the operations within the black box model are quite complicated. For example, colour information has to be extracted from a black and white image. Positional information obtained from the overhead camera is error-prone and therefore an eye-in-hand robotic visual servoing mechanism is required to make the necessary correction.

The use of a fixed overhead camera and a robot mounted camera has become an increasingly popular vision set-up in advanced robotic applications **[Hashimoto91]**, **[King88]**, **[Weiss87]**, **[Westmore91]**, **[Wijesoma93]**, **[Jang91(I)]**, **[Jang91(II)]**. In this type of robot vision arrangement, initial approximate object position is derived from the overhead camera image. As the robot reaches for the object, the robot mounted camera provides visual feedback to guide the robot towards the object with increased accuracy.

4.1 Global Image Processing

This is the first level of image processing in the system. The objective is to extract from a digitised image of the snooker table positional and colour information of the balls present. It also has to deal with situations where two or more snooker balls cluster together forming a pack. Subsequent shot determination and planning computations will be based upon data of snooker balls derived from the global image. Therefore it is vital that this module performs reliably and accurately.

4.1.1 Image Digitisation

Using the equipment described in chapter 1, the focal length of the zoom lens (i.e. variable focal length) attached to the overhead camera is set such that the snooker table fills up the entire window, as show in figure 4.1. The view area includes 20 mm of the cushion on the four edges. This may appear to be a waste of valuable pixels that do not carry any information. However, it is necessary to ensure that a snooker ball lying anywhere on the table, particularly those very near to a pocket, will not be missed. The snooker table is digitised as a matrix of 350 (x) by 146 (y) pixels by the overhead camera.

When digitising an image, each pixel in the CCD produces a voltage (ranging from 0 to 1250 mV) that is proportional to the brightness of the image at the corresponding point.

A pixel receiving no light produces a zero voltage, while a saturated pixel produces 1250 mV. This voltage is then converted into a gray-scale value, which is an integer between 0 and 63, indicating the level of light received by that particular pixel.

With the Automatix AV-5, the dynamic range of the CCD can be set according to two integer parameters 'BLACK' and 'WHITE' in a data structure called 'PICTURE SCHEDULE' where both 'BLACK' and 'WHITE' lie between 0 and 1250 and the value of 'BLACK' is always less than that of 'WHITE'. Any pixel giving a voltage less than 'BLACK' has a gray-scale value of 0. On the other hand, any pixel giving a voltage greater than 'WHITE' has a gray-scale value of 63. Any pixel giving an intermediate voltage will be assigned the corresponding gray-scale value.

Using the lighting set-up described in chapter 1, the overhead camera aperture is set at f/8, and the BLACK and WHITE settings are 0 and 1000 respectively. The WHITE level is set at 1000 rather than 1250 so that fine-tuning of the scene brightness is possible by altering the WHITE setting . Despite the high light intensity provided by the lighting grid, extreme variation of ambient lighting in the laboratory can confuse the system e.g. the yellow ball could be recognised as white if there is a sudden rise in illumination. To compensate, the WHITE level can be changed to ensure correct colour identification. Decreasing the WHITE level brightens the scene electronically which can negate the effects of a drop in illumination. Conversely, increasing the WHITE level would counter the effects of a rise in illumination.

4.1.2 Image Enhancement

Various techniques exist that serve to enhance a digitised image to suit specific applications. However, none of them seemed to offer any real improvements in the task of identifying snooker balls. One reason is that the lighting grid ensures reliable and repeatable digitisation of the scene with minimal loss or noise. Another reason is that image enhancement operations could result in undesirable effects in this application. For example, image smoothing by applying a filter actually reduces the sharpness of edges. Image sharpening by differentiation results in a loss of object gray-scale data and requires a separate copy of the image for extracting colour information.

As an example, image sharpening by differentiation using *Roberts gradient* was experimented upon. Figure 4.2 shows the procedure used which emphasises the difference in gray-scale levels between adjacent pixels. This method enhances every pixel in any digitised image except those in the right-most column and bottom row. For more details see **[Gonzalez87]**.

Figure 4.3(a) shows an image of the snooker table with all the balls on it before image sharpening was applied. Figure 4.3(b) shows the same image after the image sharpening procedure described in figure 4.2 was applied. It is clear that image sharpening has worked as expected so much so that the white, yellow and pink balls becomes a bright ring after processing. For the brown, blue, green, and black balls only the lighting reflection off the top of these balls are picked up. Red balls appear similar before and after processing. The conclusion is that this image sharpening procedure does not really enhance segmentation in any way because of the quality of the original image.

There is a computation cost associated with all image enhancement operations. In the above image sharpening procedure it actually took the Automatix AV-5 approximately 720 seconds (12 minutes) to process the image. With current technology, image processors are capable of performing enhancement of a 512 X 512, 256 gray-scale level image well within one frame time (1/30 second). Therefore, after considering the costs and gains, it is decided that image enhancement can be omitted without jeopardising the extraction of object information.

4.1.3 Image Segmentation

Segmentation is the process of recognising parts or objects that are present in a digitised image. The Automatix AV-5 running RAIL 6.04 contains built in functions that perform thresholding and connectivity analysis to identify objects in an image.

In a digitised image of the snooker table with a number of snooker balls on it, the majority of pixels will have a gray-scale level that corresponds to the brightness of the green baize. It is therefore necessary to determine a threshold so that snooker balls can be identified.

In snooker, the colour of a ball is either white, yellow, pink, red, brown, blue, green, or black. To gain a better understanding of how colours appear in black and white, 8 balls (1 for each colour) are lined up on the table as shown in figure 4.4 and the image is then digitised. Gray-scale values of the pixels around each of the 8 colour balls are tabulated in table 4.1(a) - (h). From the tabulated gray-scale values, it can be seen that the table cloth has a gray-scale value that lies predominantly between 8 and 10. This indicates that the threshold for segmentation must be greater than 10 or parts of the cloth can be mistaken as objects.

Since bright objects are to be segmented from a dark background, the system functions in the positive segmentation mode. The optimal threshold value has to satisfy the following 2 conditions :

(1) the threshold must not be so low that the area of the identified object is larger than the ball size. An excessively low threshold value results in small bits of table cloth surrounding a ball to be recognised as part of it because of the halo effect and therefore the object centre may not correspond to the true ball centre.

(2) the threshold must not be so high that the dark colour balls (i.e. BROWN, BLUE, GREEN, and BLACK) cannot be identified. This is because balls of these colours tend to blend into the green baize when digitised and can only be identified reliably and consistently by the lighting reflection off the top of them.

By experimenting with various threshold settings, the optimal threshold is found to be 19. Also, at this setting minor lighting variations can be tolerated without affecting ball identification.

Figure 4.5 shows how the 8 colour balls are outlined as objects by the AV-5. It can be seen that bright colour balls are outlined accurately. However, as already mentioned, only the lighting reflection at the top of the dark colour balls are recognised. This may not be the ideal way to recognise a ball but is nonetheless a good approximation to the true ball centre. Furthermore, owing to factors such as imperfect optics and image distortion, ball positional data determined by the overhead camera can be up to 10 mm inaccurate for all colours. Therefore it is acceptable to approximate the centre of a dark colour ball by the reflection off the top of it . In practice, this is confirmed by the fact that on average, positional data for the brown, blue, green, and black balls are no worse than that of the other colours.

4.1.4 Colour Information Extraction

At this point, the problem is half-solved. Balls can now be identified as objects but a method to determine the colour of a ball from a gray-scale image has yet to be devised. Using the pixel gray-scale data in table 4.1(a) - (h), a 3-dimensional graph showing gray-scale value at every pixel location is plotted, as shown in figure 4.6. The graph gives a very good impression of how the various colours appear in black and white as seen by the Automatix. Figure 4.7 shows the gray-scale profile of the various colours, using the same data as in figure 4.6.

Figure 4.6 and 4.7 show that the brown, blue, green, and black balls all exhibit a narrow, pointed peak in their gray-scale profile. This peak corresponds to the lighting reflection

at the top of a dark colour ball. Remember that a threshold of 19 was set, the white, yellow, pink, and red balls all have well defined edges which facilitate accurate and reliable segmentation. For the brown, blue, green, and black balls, only the top halves of the pointed peaks lie above the threshold. This means that only the lighting reflections are picked up, which agrees with what is said in the previous section.

To determine the colour of a ball, the average gray-scale value of the pixels representing the ball is computed. Table 4.1(a) - (h) is drawn deliberately with an X/Y aspect ratio that resembles the aspect ratio of a pixel. Generally speaking, pixels are actually rectangular in shape (though the use of square pixels CCDs are gaining popularity) and so its dimension in the X and Y sense are not equal, their quotient being termed the aspect ratio. Table 4.1(a) - (h) can therefore be viewed as a pixel map. Using the object features of the white ball (white is selected because its object outline fits the ball best), it is found that a snooker ball in pixel space has a dimension of 11 X-pixels by 9 Y-pixels. Although the dimension in X and Y are not equal, this matrix is indeed an approximate square because a pixel in the CCD of the overhead camera has an aspect ratio of 0.86. A circle can be drawn to fit inside this matrix, representing a ball as shown in figure 4.8.

Within this 11 × 9 pixel matrix, a set of pixels is identified. These are pixels which lie entirely within the circle in figure 4.8 and least affected by shadowing. Pixels not in this subset are ignored in the average gray-scale computation because they either lie outside of the circle or on the boundary and their pixel values are not representative of the ball's colour. Figure 4.9 (a) shows the pixels belonging to this category.

Referring back to figure 4.6 and 4.7, the profiles of the brown, blue, green, and black balls all exhibit a comparable peak above gray-scale level 16. It is therefore necessary not to use the pixels corresponding to these peaks in computing the average gray-scale value, otherwise the difference between colours will be diluted by these peaks caused by the lighting reflections. On closer inspection of table 4.1 (a) - (h), paying particular attention to the gray-scale maps of the dark colour balls, it can be seen that the reflection peaks correspond to 21 pixels around the centre of the ball, as shown in figure 4.9(b). Gray-scale profiles of the white, yellow, pink, and red balls do not show such dramatic gray-scale peaks, not because these colours are non-reflective, but because the peaks are obscured by the surrounding pixels which have similarly high gray-scale values. Therefore, for all colours, these 21 pixel values corresponding to the lighting reflection are ignored and do not contribute to the computation of the average gray-scale.

Figure 4.9(c) shows the resultant mask that is used to compute the average gray-scale value of a snooker ball recognised by the vision system. The mask centre will be placed over an object centre and the gray-scale value of the 30 unshaded pixels will be used to

compute the average gray-scale value that determines the colour of the ball. However, not every unshaded pixel is used in finding the average. Instead the 30 pixels will be sorted by their gray-scale values. Pixels with the 10 highest and 10 lowest values will be excluded, leaving 10 pixels whose gray-scale values will be averaged. This is a simple measure to reduce the possibility of random noises or lighting variations affecting the consistency of the correspondence between average gray-scale values and the various colours. Also, an object centre determined by the image processor does not always coincide exactly with the true ball centre and it is impossible to exactly define where the lighting reflection is going to lie in the pixel map. Suppose all the 30 unshaded pixels in the mask have consistent values, ignoring the values of the brightest and darkest pixels will have very little effect on the computed average. On the other hand, pixels in the mask that have unusually high or low gray-scale values will be eliminated after sorting as they will be discarded. For example, using the gray-scale data in table 4.1 (a)-(h), average gray-scale values corresponding to the 8 colours are computed in the following list :

Colour	Sample Average Gray-Scale Value
White	40.4
Yellow	36.7
Pink	31.1
Red	22.3
Brown	11.9
Blue	8.4
Green	6.1
Black	1.3

It must be pointed out here that the above values are by no means the definite mapping between average gray-scale values and colours but only a typical example. Despite every effort made to ensure uniform illumination, the average gray-scale value for a colour still varies slightly. To understand this variation better and to establish a method of classifying the colour of a ball by its average gray-scale value, a much larger sample of colour/average gray-scale data must be analysed.

The analysis involves marking one hundred points on the snooker table (10 across the length of the table and 10 from top to bottom). For each colour ball, an image of the ball at each of the 100 marked position is digitised and its average gray-scale value is computed using the method described previously. This results in 100 computed average gray-scales value per colour i.e. 800 values in total. The results are tabulated in table 4.2 (a) and (b).

Figure 4.10 is produced by plotting the average gray-scale values of each colour against the trial number. The data can also be expressed as a frequency histogram for average gray-scale values as in figure 4.11. Figure 4.10 and 4.11 show that the ranges of average gray-scale for the 8 colours never intersect. This is an important observation because it shows that in this sample, any individual can only be mapped to a single colour and therefore no confusion can arise. However it can be seen that the green and blue data are very close at some points. Indeed the difference between the maximum green and minimum blue average gray-scale values is only 0.6 gray-scale. All other colours are well separated. This indicates that the most likely mistake in colour recognition is that of identifying a green ball as blue and vice versa. The following analysis will prove statistically that the chance of this mistake occurring is negligible.

In figure 4.11 most colours have a frequency histogram that resembles a normal distribution. The only exception being the green colour which has a particularly high frequency at an average gray-scale value of 7.0. Nevertheless, since 7 out of 8 colours have an average gray-scale histogram that fits a normal distribution, it can be generalised that for each colour, the average gray-scale value is normally distributed with its mean and standard deviation as listed in table 4.3.

To determine the probability of mistaking an object's colour, the normal distribution curves of the 8 colours are drawn. Figure 4.12 contains distribution curves for black, green, blue, and brown, figure 4.13 contains that for red, pink, yellow, and white. For each curve, only the portion that lies between plus or minus 3 standard deviation from the mean is drawn. In a normally distributed population the probability of any one sample lying outside ± 3 SD is 0.0026, which is a really minute chance. Figure 4.12 and 4.13 show that none of the distribution curves crosses another, meaning that there is less than a 0.0026 chance of the average gray-scale value of a colour can lying within 3 SD of a neighbour colour.

Finally, a classifier system to infer the colour of an object using its average gray-scale value is devised. For each colour, low and high thresholds are established such that an object with an average gray-scale value lying between the low and high threshold of a colour will be classified as having that colour. Where possible, the low and high thresholds of a colour are set as its mean gray-scale - 5 and + 5 times its standard deviation respectively to ensure that colour thresholds are well separated. In case the statistical distance between the means of two neighbouring colours is not wide enough to accommodate 5 times the sum of their standard deviation (e.g. green and blue), a point that is equi-distant from the 2 means concerned in terms of standard deviations will be

selected as the boundary. The low and high threshold selection algorithm is presented below :

Let Colour1 and Colour2 be 2 adjacent colours in figure 4.12 and 4.13

And Colour1 has mean μ_1 and standard deviation s_1
Colour2 has mean μ_2 and standard deviation s_2
$\mu_1 < \mu_2$

Then

$$\text{High Threshold of Colour1} = \mu_1 + \text{MIN}\ (\ 5,\ (\mu_2-\mu_1)/(s_1+s_2)\ )\ *\ s_1$$

$$\text{Low Threshold of Colour2} = \mu_2 - \text{MIN}\ (\ 5,\ (\mu_2-\mu_1)/(s_1+s_2)\ )\ *\ s_2$$

This algorithm is applied to every adjacent pair of colours to determine the low and high thresholds of each colour. The colours black and white are border cases and are taken care of as follows :

$$\text{Low Threshold of Black} = \mu_{BLACK} - 5\ *\ s_{BLACK}$$

$$\text{High Threshold of White} = \mu_{WHITE} + 5\ *\ s_{WHITE}$$

The threshold settings for the 8 colours are listed in table 4.4. Suppose the average gray-scale value of an object is computed as $GRAY_{obj}$. Then the following inequalities are checked :

$$LOW_C <= GRAY_{obj} \text{ and } GRAY_{obj} < HIGH_C$$

where LOW_C = Low Threshold of Colour C

$HIGH_C$ = High Threshold of Colour C

C = {black, green, blue, brown, red, pink, yellow, white}

Since the threshold ranges of the colours are disjoint, the inequalities can only hold true for a single colour, and the object will be classed as having that colour. If an object's average gray-scale does not fall between any 1 of the 8 threshold ranges, it will be classified as having an unknown colour and an error will be reported.

4.1.5 Conversion From Pixel Co-ordinates To World Co-ordinates

Having solved the problem of identifying the colour of a snooker ball, its position in world co-ordinates has to be determined. To do this, the various co-ordinate systems involved have to be identified.

To begin with, objects identified by the Automatix image processor have their associated attributes computed automatically. These attributes include centre of the object and other geometric properties, all in pixel co-ordinates. The origin of the pixel co-ordinate system is at the top-left corner of the digitised image, with the positive x-axis running across the image from left to right and the positive y-axis from top to bottom.

The world co-ordinate system is central to the whole system and all other co-ordinate systems are related to it. Units in the world co-ordinate system are metric and the system works in millimetres. The origin of the world co-ordinate system is defined as the top-left corner of the overhead camera view window, as shown in figure 4.1. Axes of the world co-ordinate system have the same orientation as that of the pixel co-ordinate system of the overhead camera.

According to the definition of the view window, the top edge of the window is defined by the line y = 57 and the left edge is defined by the line x = 3, both in pixel co-ordinates. Hence the origin of the world co-ordinate system is at the point (3,57) in pixel co-ordinates. It can thus be deduced that the 3D homogeneous transformation from pixel to world co-ordinates is :

$$ {}^{world}T_{pixel} = \begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -57 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{Eq. } \textit{(4.1)} $$

To transform a point from pixel co-ordinate to world co-ordinate, the following transformation equation is applied :

$$^{world}\mathbf{P} = {}^{world}\mathbf{T}_{pixel}\,{}^{pixel}\mathbf{P} \qquad \textbf{Eq. } \textit{(4.2)}$$

where $^{world}\mathbf{P}$ = 4 × 1 position vector P in world co-ordinates in pixel scale

$^{pixel}\mathbf{P}$ = 4 × 1 position vector P in pixel co-ordinates in pixel scale

$^{world}\mathbf{T}_{pixel}$ = 4 × 4 homogeneous transformation from pixel co-ordinates to world co-ordinates

By multiplying out Eq. *(4.2)*, the following is obtained :

$$^{world}\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -57 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{pixel}P_x \\ {}^{pixel}P_y \\ {}^{pixel}P_z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} {}^{pixel}P_x - 3 \\ {}^{pixel}P_y - 57 \\ {}^{pixel}P_z \\ 1 \end{bmatrix}$$

$$= {}^{pixel}\mathbf{P} + \begin{bmatrix} -3 \\ -57 \\ 0 \\ 0 \end{bmatrix} \qquad \textbf{Eq. } \textit{(4.3)}$$

This simple relation between the 2 co-ordinate systems is a result of the fact that the axes in both systems have identical orientations, thus requiring no rotational transformation.

In this application only transformations in 2D are dealt with, therefore all z-components in positional vectors and transformations can be omitted. Thus the following is the effective relationship between the pixel and world co-ordinate systems :

$$\begin{bmatrix} {}^{world}P_x \\ {}^{world}P_y \end{bmatrix} = \begin{bmatrix} {}^{pixel}P_x \\ {}^{pixel}P_y \end{bmatrix} + \begin{bmatrix} -3 \\ -57 \end{bmatrix} \qquad \textbf{Eq. } \textit{(4.4)}$$

To complete the conversion from pixel to world co-ordinates, the relationship between the dimension of the view window in pixel space and the actual physical dimension of the window in millimetres has to be established. The view window is composed of a pixel matrix of 349 X-pixels by 145 Y-pixels. The physical size of the view window is measured as 1560 mm across and 768 mm top to bottom, resulting in the following transformation :

$$\begin{aligned} 349(\text{Xpixel}) &\equiv 1560mm \\ 145(\text{Ypixel}) &\equiv 768mm \end{aligned} \Rightarrow \begin{aligned} (\text{Xpixel}) &\equiv 4.47mm \\ (\text{Ypixel}) &\equiv 5.3mm \end{aligned}$$

Note that a pixel has an aspect ratio of 0.862938 and therefore the x and y pixel scale factors are not equal. Other methods to determine the scale factor can be found in **[Penna91]** and **[Lenz88]**. The method described here is admittedly rudimentary yet it does allow the overhead camera to be calibrated rapidly giving acceptable accuracy. Any positional error due to image distortion or calibration will be corrected by the visual servoing operation described later.

Having determined the scaling factors, the conversion function can be formulated as Eq. *(4.5)* below :

$$\begin{bmatrix} {}^{world}P_{x_{mm}} \\ {}^{world}P_{y_{mm}} \end{bmatrix} = \begin{bmatrix} 4.47 & 0 \\ 0 & 5.3 \end{bmatrix} \left(\begin{bmatrix} {}^{pixel}P_x \\ {}^{pixel}P_y \end{bmatrix} + \begin{bmatrix} -3 \\ -57 \end{bmatrix} \right) \qquad \textbf{Eq. } \textit{(4.5)}$$

where ${}^{world}P_{x_{mm}}$ = X co-ordinate of point P in world co-ordinate system in millimetres

${}^{world}P_{y_{mm}}$ = Y co-ordinate of point P in world co-ordinate system in millimetres

There is one last complication in the conversion function though. The X co-ordinate of the centre of an object identified by the Automatix is scaled. Therefore, before the X co-ordinate of the centre of an object can be fed into Eq. *(4.5)* for conversion, it has to be divided by the aspect ratio, which equals 0.862938 for the overhead camera.

4.1.6 Data Structure for Storing Ball Positions

Ball positions in world co-ordinates in millimetre scale are stored in 2 data structures : a 10 by 2 REAL array named 'REDPOS' to store the (x, y) co-ordinates of the 10 red balls and another 8 by 2 REAL array named 'COLOURPOS' to store the co-ordinates of the other 7 colour balls. Figure 4.14 is a list of constants where each colour is declared as a constant term of an integer value equivalent to the score it carries, with the exception of the cue (white) ball which has an arbitrary value of 8. The structures of the 2 position arrays are shown in figure 4.15 and 4.16.

Initially all array locations hold a rogue value, -1000, indicating that no ball has been identified. If the colour of an object is identified as RED, its positions will be recorded into the array 'REDPOS'. Note that the array 'REDPOS' is filled up sequentially starting at row index 1. An integer variable 'NRED' keeps track of the number of red balls identified. 'NRED' is initialised with the integer value 0 since no red ball exists before an image is processed. On identifying a red ball, 'NRED' is incremented by 1 and the position of the ball is stored in the array 'REDPOS' at row 'NRED'. If a red ball is identified when 'NRED' already equals 10 (the maximum number of red balls), an error will be reported.

Unlike red balls, there can only be one ball having each of the colour yellow, green, brown, blue, pink, black, and white. Therefore on identifying a non-red colour ball, duplicate entry must be prevented. Suppose a ball is identified as green, then the contents of the array 'COLOURPOS' at row 'GREEN' are checked. Remember that all array contents are initialised to have the rogue value of -1000, non-negative contents at row 'GREEN' would indicate that the green ball is already identified and so a duplicate entry is detected, resulting in an error message. Otherwise, the position of the newly identified green ball would be stored in the array 'COLOURPOS' at row 'GREEN'. This duplicate check is carried out each time a non-red colour ball is identified.

4.1.7 Identifying Balls in a Pack

In snooker, it is not unusual to find several snooker balls clustering together to form a pack. The most trivial case is found at the beginning of a frame when all 10 red balls (15

in a full-size snooker table) are placed together to form a triangular pack. When 2 or more balls cluster together, the aforementioned algorithm for identifying snooker balls fails because it is designed to deal with isolated snooker balls. What happens is that a cluster of balls would be recognised by the image processor as a single large object, which is clearly wrong.

To solve the problem, 3 distinct operations are required : (i) detection of existence of a pack of 2 or more balls, (ii) recording of the relevant data about a pack, and (iii) separation and identification of each constituent ball in a pack.

To determine whether an object is a single snooker ball or a pack of several balls, two object features have to be looked at. The first is the 'AREA' feature which is the area of the identified object in pixel scale. The second is the 'PEROUND' feature which is a measure of the roundness of the object. Inspection of these 2 features of every colour ball reveals that for a single snooker ball of any colour, the 'AREA' is always less than 80 and that the 'PEROUND' is always greater than 0.65. Thus any object with 'AREA' ≥ 80 and 'PEROUND' ≤ 0.65 will be regarded as a pack of 2 or more snooker balls. Figure 4.17 show a sample pack of 5 red balls.

On the snooker table, there can be a maximum of 17 snooker balls : 10 reds, 6 colours and the cue ball. However, not every single ball is capable of merging with another ball in the digitised image to form a pack. The exceptions are the black, green, blue, and brown colour balls because only the lighting reflections on top of these balls are recognised and this reflection is always well separated from any neighbouring or even touching ball (see figure 4.6). Therefore only 13 snooker balls can merge with each other to form packs of snooker balls. What is needed next is a way of storing the data about a pack. As more packs are identified, a list of pack information is maintained. Once all packs are identified, this list will be processed to retrieve the constituent balls within each pack.

The pack information stored are the 4 extremities of the pack object i.e. the minimum and maximum x and y values of the object in pixel co-ordinates. These 4 values define the smallest window in which the pack object fits. When it comes to resolving the pack, processing will be focused upon the digitised image inside this pack window. Since only 13 snooker balls can forms packs, there can exist at most 6 packs. A 6 by 4 integer array 'PACK_LIST' is created to store pack information with an integer counter 'PACK_NO' to keep a record of the number of packs identified. This array can be viewed as a list of windows each containing a pack. An error message will be reported if more than 6 packs are identified.

Finally the task of identifying individual balls within a pack has to be tackled. To resolve a general pack of *n* snooker balls, the simplest case of a pack of 2 snooker balls is first investigated. With the gray-scale threshold set at 19, a pack of 2 balls will always appear as a single object. By increasing the threshold at increments of 1 and performing connectivity analysis this object can be gradually separated into 2. Table 4.5 (a) is a pixel map of a pack of 2 red balls at a threshold setting of 19. Pixels with gray-scale value above the threshold (i.e. 20 or above) are shaded and they form an object. In this particular example, the entry in 'PACK_LIST' corresponding to the pack is [143,134,158,146] (i.e. min. X = 143, min. Y = 134, max. X = 158, and max. Y = 146), and this window is highlighted in the table. The threshold is then increased at increments of 1 gray-scale and the results are shown in Table 4.5 (b), (c), (d), and (e) where the threshold is set at 20, 21, 22, and 23 respectively. Note that Table 4.5 (a)-(e) depicts the same pixel data, but the objects identified are different because of the changing threshold setting. At a threshold of 23, the original object is separated into the 2 balls and the pack is resolved since no object falls into the category of a pack as already defined above. In this example, 5 iteration of pack resolving procedure has taken place. Each iteration involves digitising the image, and every object identified in the pack window is checked to determine whether it is a pack. This algorithm can be used to resolve a pack containing *n* balls. It is summarised by the pseudo-code in figure 4.18.

4.2 Eye-in-Hand Visual Servoing

As described in chapter 4.1 positional information of snooker balls are determined by the overhead camera. Because of the distance between the camera and the table surface, the wide angle of view, and imperfect lens optics, image distortion is inevitable and positional information derived from a distorted image is bound to be inaccurate. To calculate the shot path and cue angle for a successful shot, accurate positions of the cue ball and target ball are most important (positions of the pockets are pre-determined and do not change as the overhead camera is fixed in relation to the snooker table). Thus a mechanism of correcting this positional error is crucial to successful potting.

In a conventional servo mechanism, positional feedback is provided by an encoder of some sort (e.g. optical, mechanical, or magnetic). Basically speaking, the servo mechanism will command the drive to effect a change (which can be electronic or mechanical) until the encoder reading matches that required. This mechanism can be applied in the system to correct overhead camera error with some modifications. The key component of the system is a sensor whose sole purpose is to provide positional feedback. After much consideration, it is decided that an on-board CCD camera would serve this purpose satisfactorily.

4.2.1 On-board Camera Installation

In choosing a suitable position for installing the on-board camera, there are 2 major considerations. First of all, like the pneumatic snooker cue, the on-board camera must be able to reach any point on the snooker table. This in turn means that the work envelope of the on-board camera must be maximised as much as possible, bearing in mind that the SKF Linear Driver is only capable of positioning the PUMA at any 1 of 7 fixed positions plus a 'HOME' position. Secondly, determination of the on-board camera position with respect to the robot base should not involve separate inverse kinematics computation. Suppose the on-board camera is directly fitted to link 3 of the PUMA 560 (which corresponds to the human lower arm), then a new set of link parameters will have to be determined. The degree of freedom of the PUMA arm with respect to the on-board camera would effectively be reduced from 6 to 3, which significantly reduces the flexibility and manoeuvrability of the on-board camera. Also determination of link parameters is error-prone and additional processing would be required to work out the camera position in PUMA base frame involving complex matrix operations.

To satisfy the above requirements, a decision is made to mount the on-board camera to the tool holder to which the pneumatic cue is also fitted. Thus the PUMA robot arm actually carries 2 pieces of tools all the time. The close proximity between the 2 tools also means that their respective work envelopes are near identical.

The available space for fitting the on-board camera and lighting is extremely confined. The physical size of the robot wrist assembly dictates that nothing mounted to the wrist should protrude beyond the tool mounting flange towards the robot base in the PUMA 'READY' position. Otherwise collision between any protruding part of the tool and the robot wrist will occur as joint 6 is rotated. Another restriction is that the on-board camera and lighting assembly cannot protrude beyond the pneumatic cue along the axis of joint 6 away from the robot base. This is to prevent collision between the camera and the snooker table surface when the cue is lined up horizontally to take a shot.

In order to conform to the above restrictions, whatever on-board camera and lighting assembly installed must not exceed 95 mm in length. Also, care should be taken not to exceed the 4.0 kg payload of the PUMA robot arm. Unfortunately, no conventional CCD camera is small enough to satisfy the constraints. To get round the problem, a remote head CCD camera is used. With this type of camera, the CCD and lens are fitted in a separate housing detached from the main camera electronics. The advantages of having a remote head are reduced weight and ease of installation where space is confined. The digitised image is transmitted from the remote head to the main camera body via a cable.

A Pulnix TM540 remote head CCD camera was purchased after considering its cost and specification. It is then configured into the Automatix system with a camera port and pixel buffer allocated. A 16 mm lens is fitted to the remote head which together measures 94 mm in length, just within the size limit. Lens focusing is of the internal type which means that the external length of the lens stays constant as the focus is altered.

Uniform illumination is critical to the performance of the visual servoing mechanism. While visual servoing is in operation, the PUMA robot arm will be positioned over the snooker table by the SKF driver thus casting a shadow over the snooker ball being examined. The shape and size of the shadow depends on the posture of the PUMA robot arm and is unpredictable. Reliable visual servoing is not possible under such circumstances and so auxiliary lighting must be incorporated to ensure a uniform level of illumination. Ideally, 4 light sources fitted around the lens would illuminate the scene uniformly with minimal shadowing. But because of the space limitation, only 2 miniature metal halide reflectors can be fitted, which is an acceptable compromise. Indeed the 2 reflectors used produce more than adequate lighting so much so that two 2X neutral density filters have to be fitted with the lens aperture set at approximately f/16 to prevent saturation of the CCD. Figure 4.19 is a photograph of the complete tool assembly.

4.2.2 Mechanism of Visual Servoing

Like the overhead camera, the on-board camera is to operate at a fixed height from the snooker table surface. The on-board camera is carefully installed to ensure that the optical axis of the lens is parallel to the PUMA Z-axis in order to minimise parallax error. The on-board lighting is also set up to uniformly illuminate a scene at such a distance. In selecting the operating distance, there are two contrasting requirements. On the one hand the on-board camera is to be positioned as close as possible to the snooker table surface so that a snooker ball would fill up the field of view to gain the maximum resolution. On the other hand, it should be positioned as far away from the table surface as possible for more uniform illumination and less shadowing. A compromise is struck by positioning the camera at a height such that the distance between the tool mounting flange and the centre of a snooker ball being viewed equals 297 mm. Figure 4.20 is a typical image of a snooker ball produced by the on-board camera.

Visual servoing has been successfully applied in many tasks related to robotics. Bowman and Forrest **[Bowman88]** demonstrated in their work that visual servoing can be used as a tool to determine machine accuracy and repeatability apart from its obvious application in detecting differential movement of an object. Allen, Yoshimi, and Timcenko **[Allen91]** built a real-time visual servoing system enabling a robot arm to track a moving

object. In that system, two static cameras were used to produce 3D visual information. In the work of J. Y. Zheng, Q. Chen, and S. Tsuji **[Zheng91]**, a more versatile form of visual servoing was implemented where the camera in the system can be actively guided to produce images from various viewing distances and view points to improve system robustness. In general, a visual servoing system can involve a single camera for 2D or multiple cameras for 3D visual feedback. Also, camera installation can be static or mobile (by being attached to a robot, usually at close proximity to the end-effector).

At this point it must be remembered that the need for visual servoing arises from the fact that positional data of any snooker ball computed by the overhead camera is accurate to within 10 mm only, and the positional error has to be corrected by servoing the on-board camera to locate the ball to a higher accuracy. The detail mechanism of commanding the PUMA robot arm to position the on-board camera at the chosen operating height over a particular snooker ball is covered in chapter 2 and so it is not repeated here. Instead the detection of the positional error and the iterative servoing algorithm that is used to locate the ball more accurately will be concentrated upon.

The task of visual servoing in locating snooker balls has the following characteristics :

(1) all objects to be dealt with are snooker balls

(2) the optical axis is normal to the snooker table surface and so all objects appear as circles in the image

(3) orthogonality between the scene and optical axis is maintained at all times

(4) only the (x, y) positions of snooker balls on the snooker table surface are relevant hence a 2D visual servoing system is required

Note that points (2) and (3) above are also stated as prerequisites of good servo performance in **[Westmore91]** where a similar Eye-in-Hand servoing set-up was devised.

A position-based control strategy has been adopted. The characteristics of this approach is the sequential 'look' operation of the robot-mounted camera followed by the 'move' operation of the robot arm. The required move in robot Cartesian space is derived from the expected and actual position of the object feature in image space. The feature of interest is the centre of the snooker ball being tracked. The disadvantage of position-based control is that very often several 'look-and-move' iterations are required before the object feature is located, and noisy conditions could result in instability. During 'look-

and-move' iterations, the camera must wait for the robot arm to finish its move before a new image can be acquired, making it unsuitable to track moving objects. In this application, time is not a critical factor (the system normally completes servo location of a snooker ball in under 10 seconds) and only stationary snooker balls are to be located by servoing. Together with good a priori knowledge of the image and object feature, a simple and effective position-based visual servoing mechanism can be designed and implemented.

The process of locating a ball by visual servoing can be broken down into the following sub-processes.

(I) Image Digitisation

Similar to the digitisation of the global image, the quality of the digitised image is a function of the illumination and the setting of the BLACK and WHITE levels. The level of illumination is fixed, leaving the BLACK and WHITE levels as variables. Since an individual ball is operated upon at a time, which can have 1 of 8 colours, it is best to determine the ideal BLACK and WHITE level settings for each colour. By altering the BLACK and WHITE levels and studying the effect on the digitised image for each colour, a list of settings for the 8 colours can be drawn, as shown in table 4.6.

(II) Image Segmentation

Segmentation of the on-board camera image is similar to that of the global image, but with 3 major differences.

Firstly, each colour has its own threshold level for segmentation owing to their differences in gray-scale contents and the BLACK and WHITE level settings.

Secondly, each colour in fact has 2 threshold levels. Because of the error in the position of a ball, the first digitised image is likely to show a ball that is not at the centre of the on-board camera frame. If only 1 threshold level is used to segment a ball, then it is possible for the image processor to wrongly identify the table cloth, which lies in the central region of the on-board lighting, as an object. To prevent this vision error from occurring, it is necessary to use a high threshold so that a ball can still be identified when lying off-centred in the camera view window. The high threshold is used to segment the digitised image until the deviation between the object centre and the reference point is less than 5 mm.

When that happens, the threshold is replaced by a lower value which allows the ball to be accurately segmented.

Thirdly, the black ball is segmented as a negative object i.e. pixels in the digitised image having gray-scale values below the threshold form the object. All other colours are segmented in the positive mode where pixels with gray-scale values above the threshold form the object.

Finally, in the on-board camera digitised image, only the most dominant object is identified. In other words, segmentation of the on-board camera image results in only 1 object which has the biggest area in case of multiple objects. Table 4.7 is a list of the high and low threshold settings and the mode of segmentation.

(III) Computation of Deviation Between the Expected and Actual Ball Centres

The expected ball centre (X_{exp}^{scaled} , Y_{exp}) is defined as the centre point of the on-board camera view window. It is common practice to scale the X pixel size to match that of the Y pixel size and so the Y pixel is always understood to be unscaled. This point is defined as (373, 268) in scaled X and Y pixel co-ordinates. The actual ball centre (X_{actual}^{scaled} , Y_{actual}) is the centre of the object resulting from segmentation of the on-board camera image. Note that both X_{exp}^{scaled} and X_{actual}^{scaled} are scaled which means that they have to be divided by the aspect ratio of the on-board camera pixel to be converted into unscaled pixel co-ordinates in Step (IV). The X and Y deviations in scaled pixel space are computed as :

$$d_x^{scaled} = X_{actual}^{scaled} - X_{exp}^{scaled}$$

$$d_y = Y_{actual} - Y_{exp}$$

where $X_{exp}^{scaled} = 373$ and $Y_{exp} = 268$

(IV) Converting the Deviation From Pixel Scale to World Scale

This process is also commonly known as camera calibration. This process is greatly simplified in this application because of the pre-requisite conditions of orthogonality between optical axis and scene and a constant viewing distance. The X and Y pixel scales are determined by actually measuring the physical size

of the on-board camera view window. The on-board camera is positioned at the operating height from the table surface and a piece of metric scale graph paper is placed underneath the camera and an image is taken. Care has to be taken to place the piece of paper 22 mm (approx. radius of a snooker ball) above the table surface since the lens is set to focus at this level. Figure 4.21 shows this image of the graph paper with the view window superimposed. It shows that the view window measures 53 mm by 52 mm in the X and Y direction respectively.

The size of the camera view window in pixel scale is also known since it is defined to enclosed a snooker ball with some margin to spare, as shown in figure 4.22. The dimension of the view window is 165 unscaled X-pixels by 281 Y-pixels.

Transformation between world scale and on-board camera pixel scale can now be established. By dividing the physical measurement by the corresponding number of pixels, an unscaled X pixel is found to be equivalent to 0.32 mm and a Y pixel equivalent to 0.19 mm.

Let (d_x^{mm}, d_y^{mm}) = deviation between the expected and actual centres of a snooker ball in millimetre scale

Then $d_x^{mm} = 0.32 (d_x^{unscaled})$mm **Eq. *(4.6)***

$d_y^{mm} = 0.19 (d_y)$mm **Eq. *(4.7)***

where $d_x^{unscaled} = d_x^{scaled} / 1.70029$

1.70029 being the aspect ratio of the on-board camera pixel

(V) Commanding the PUMA to eliminate the deviation (d_x^{mm}, d_y^{mm})

Before a motion command can be issued to the PUMA, it is necessary to understand the orientation relationship between the on-board camera frame and the PUMA co-ordinate frame. As already mentioned the optical axis of the on-board camera is always orthogonal to the X-Y plane of the world co-ordinate system, i.e. the snooker table surface. In other words, the on-board camera X-Y plane is always parallel to the world X-Y plane. This allows 1 degree-of-freedom for the orientation of the on-board camera frame with respect to the world frame.

To keep the system simple without compromising its effectiveness, this degree of orientation freedom is eliminated by fixing the orientation of the on-board camera during visual servoing.

Intuitively, the on-board camera should attain an orientation such that the on-board camera frame and the world frame have identical orientation. Since the PUMA base frame also has its orientation identical to that of the world frame, the PUMA arm can simply be commanded to make a straight line relative move of displacement (d_x^{mm}, d_y^{mm}) in the X and Y direction respectively to drive the on-board camera reference point (X_{exp}^{scaled} , Y_{exp}) towards the actual ball centre (X_{actual}^{scaled} , Y_{actual}).

In reality, the on-board camera is operated in an orientation such that the on-board camera X-axis is aligned with the world and PUMA X-axis but pointing in the opposite direction. There is no adverse effect whatsoever in performance operating the on-board camera in this orientation. This change is made purely because it is more convenient to made minor adjustments to the lens aperture and on-board lighting (which may be necessary due to the vibration associated with the firing of the pneumatic cue) with the on-board camera in this orientation. Figure 4.23 shows the relationship between the on-board camera and PUMA frames in plan view. With the defined relationship between the 2 frames, the 2-D transformation to convert (d_x^{mm}, d_y^{mm}) from on-board camera frame to PUMA base frame is :

$$ {}^{\text{Puma}}T_{\text{camera}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} $$

where θ = angle of rotation about Z-axis

$$ = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{since } \theta = 180 \text{ degrees} $$

Therefore the resultant PUMA straight line relative motion in the X and Y direction respectively equals :

$$ {}^{\text{Puma}}T_{\text{camera}} \begin{bmatrix} d_x^{mm} \\ d_y^{mm} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} d_x^{mm} \\ d_y^{mm} \end{bmatrix} = \begin{bmatrix} -d_x^{mm} \\ -d_y^{mm} \end{bmatrix} $$

Each time the PUMA arm is commanded to make such a move, the co-ordinates of the ball concerned is updated accordingly.

Let co-ordinates of the ball concerned before servo motion = (x_1, y_1)

After servo motion, the ball co-ordinates are updated as (x_2, y_2)

where $x_2 = x_1 - d_x^{mm}$

$y_2 = y_1 - d_y^{mm}$

(VI) Iterate from step (I) to (V) until (d_x^{mm}, d_y^{mm}) is below a certain threshold

It is necessary to execute step (I) to (V) iteratively for the following reasons :

1. With the target ball not centred in the on-board camera view window, which is usually the case because of overhead camera vision error, the ball does not lie within the central region of the on-board lighting. This results in parallax error and shadowing of the ball and the centre of the segmented object would not correspond to the true centre of the ball. By iteratively driving the PUMA to locate the true ball centre, the ball falls into the focus of the on-board lighting, thus allowing the ball centre to be more accurately located.

2. Eq. *(4.6)* and *(4.7)* which convert the deviation between expected and actual ball centres from pixel scale to world scale are by no means absolutely accurate. In fact it is practically impossible to determine the absolute correlation between pixel scale and world scale because of quantization error upon image digitisation. Therefore (d_x^{mm}, d_y^{mm}) can only be treated as approximations to the true deviation. Nevertheless, the visual servoing system can accommodate small errors in (d_x^{mm}, d_y^{mm}) as the functioning of the system relies on visual feedback. As the magnitude of the deviation (d_x^{mm}, d_y^{mm}) decreases at each iteration, the error term converges to zero since it is only a small fraction of the deviation. Hence the iterative process actually eliminates the small error in the scaling conversion.

Finally, as in any iterative process, there has to be a terminating condition. In this application, the best possible performance of the on-board camera vision set-up is to be extracted because the accuracy of ball positions has a direct bearing on the

success or failure of a shot. There has never been any intention to study optics and the hardware of CCD imaging devices in this research. Therefore no attempt will be made to investigate resolution related topics such as lens aberration and the modulation transfer function of the system. Instead, a practical approach is taken to find an approximation to the best possible resolution of the on-board vision system.

With the on-board camera set-up as already described, the mapping between image space and world space is established in Step (IV) above where the X pixel size is equivalent to 0.32 mm in world space and the Y pixel size to 0.19 mm. Note that this mapping itself is an approximation. With such a mapping, it is possible to visualise a rectangle K of dimension 0.32 × 0.19 mm in world space falling on to pixel PIX on the CCD through the lens.

Now imagine a point P that is imaged on to a pixel, as shown in figure 4.24 (a). Provided that point P lies within rectangle K, point P will always be imaged on to pixel PIX on the CCD. By treating the centre of rectangle K as the Cartesian origin, as in figure 4.24 (b), which shows that the on-board vision system cannot resolve better than $0.32/2$ mm in the X direction and $0.19/2$ mm in the Y direction in world space.

The above example is indeed analogous to Step (III) and (IV). In other words, the iterative process of reducing the deviation (d_x^{mm}, d_y^{mm}) is ultimately accurate to within 1 pixel in theory. In practice, using such tight tolerances sometimes results in prolonged 'hunting' of the servoing system due to excessive iterations. Therefore the accuracy constraint can be relaxed by doubling the X and Y tolerances with noticeable gain in performance in terms of the time taken to locate a ball by visual servoing. It follows that visual servoing should stop when the conditions below are true :

$$|d_x^{mm}| \leq 0.32$$

$$|d_y^{mm}| \leq 0.19$$

After termination of visual servoing, the PUMA remains in the same posture, which is preferable to driving the PUMA home after each servo operation. The total amount of machine motion will be reduced and the PUMA is ready to be commanded to perform one of three possible tasks after a servo operation : to perform visual servoing on another ball, to take a shot, or to move to the 'HOME' position. In practice, it takes

approximately 30 seconds for the system to complete visual servoing of a snooker ball, with the PUMA starting at the 'HOME' position.

4.2.3 Determination of On-Board Camera Tool Configuration

In the visual servoing system, the on-board camera has effectively become an extension to the 6^{th} joint of the PUMA arm. Therefore it is necessary to attach a co-ordinate frame to the on-board camera reference point and accurately determine the position and orientation of this frame relative to the default PUMA 'NULL' tool. This process is commonly known as Hand-to-Eye Calibration **[Bowman87]**, **[Shiu89]**, **[Tsai87]**, **[Tsai88]**, **[Tsai89]** , see figure 4.25, which enables a new tool 'CAMTOOL' to be defined that is associated with the on-board camera. The 'NULL' tool is defined by default as the centre of the PUMA tool mounting flange and its co-ordinate axes are parallel to those of the 6^{th} PUMA joint frame. The definition of the 'NULL' tool transformation is (0, 0, 0, 90, -90, 0). The on-board camera reference point, defined at (373, 268) in on-board camera pixel space, is the central pixel in the CCD chip so that optimal image quality can be obtained when the centre of the ball being viewed coincides with the reference point.

Note that the functioning of the visual servoing mechanism is inherently independent of the on-board camera tool definition. Provided an object (i.e. a snooker ball) is identified, the visual feedback mechanism will direct the robot arm to iteratively reduce any positional deviation until the reference point coincides with the ball centre. Despite successful servo operation, the updated ball position after visual servoing would be incorrect if the tool definition is not accurately determined. This is analogous to the problem arising from the use of incorrect link parameters in robot kinematics which results in poor robot accuracy.

Therefore the problem is one of finding the deviation of the on-board camera reference point (373, 268) in pixel space from the 'NULL' tool point. The orientation of the on-board camera tool 'CAMTOOL' is defined as equivalent to the 'NULL' tool. The Z-component of 'CAMTOOL' can be arbitrarily set to zero because of the a priori knowledge that the on-board camera is to function in plane parallel to the PUMA X-Y plane where Z=293 and that all objects to be identified appear as circles in 2-dimension. The definition of 'CAMTOOL' is thus (X_C, Y_C, 0, 90, -90, 0), and the values of X_C and Y_C are to be determined by the method outlined below.

Figure 4.26 (a) and (b) show the positioning of the on-board camera in the PUMA wrist assembly with the on-board camera attached. For clarity, the two on-board reflectors are not shown. It can be seen that the line *L1*, which passes through the on-board camera

reference point and is parallel to the 'NULL' tool Z-axis, is offset from the 'NULL' tool point by X_C and Y_C in the X and Y direction respectively. Figure 4.27 shows that when the on-board camera reference point coincides with the centre of a snooker ball, the position of the ball is also offset from the 'NULL' tool point by (X_C, Y_C), the z-offset being irrelevant because of the pre-condition that the optical axis is normal to the scene during operation.

To directly measure the offset (X_C, Y_C) is physically impossible because the on-board camera reference point at (373, 268) in pixel space is a photo-sensitive site on the CCD imaging chip inside the camera. Therefore an indirect method involving the imaging of a snooker ball is devised.

Step (1) Select the PUMA 'NULL' tool by issuing the VAL-II command 'tool' without any argument.

Step (2) The PUMA robot arm and the SKF Linear Driver are driven manually to position the on-board camera over a ball on the snooker table. The position of the SKF Linear Driver and the ball must not be changed until the end of the tool calibration procedure. It is important to ensure that the z co-ordinate of the PUMA 'NULL' tool equals 273 (which specifies the operating height of the on-board camera) and that the orientation equals (90, -90, 0), i.e. the 'NULL' tool X and Y axes are parallel to the PUMA co-ordinates X and Y axes.

Step (3) By inspecting the digitised on-board camera image of the ball, shift the on-board camera manually in the X-Y plane until the ball centre coincides with the reference point. This is achieved by drawing a cross in the pixel buffer at the point corresponding to the reference point which is already defined as the point (373, 268) in pixel space. The centre of the object (i.e. the ball) is computed by the Automatix built-in segmentation routine, and marked by a cross. Image digitisation and manual shifting of the on-board camera is repeated until the two crosses overlap each other, which really is the goal of the servo motion. The position of the PUMA 'NULL' tool is recorded as *P1* which equals (*x1*, *y1*, 273, 90, -90, 0) when the goal state is reached, as shown in figure 4.28.

Step (4) Repeat Step (2) & (3) with the PUMA 'NULL' tool in orientation (-90, -90, 0). This will result in the PUMA 'NULL' tool arriving at the point *P2* which equals (*x2*, *y2*, 273, -90, -90, 0), as shown in figure 4.29. Note that the ball position must not be altered when repeating Step (2) & (3).

Step (5) The position of the reference point *RP* in figure 4.28 & 4.29 relative to the 'NULL' tool origin is constant. Hence *P2* can be viewed as the resultant point of rotating *P1* about *RP* by 180^{o} and vice versa. This shows that the points *P1*, *RP*, and *P2* are co-linear and that *RP* is the mid-point of line segment *P1P2*.

Step (6) The deviation of the on-board camera reference point *RP* from the 'NULL' tool can therefore be deduced as $X_C = (x2-x1)/2$ and $Y_C = (y2-y1)/2$ in the 'NULL' tool X and Y direction respectively, as shown in figure 4.30.

To minimise experimental error and the effect of robot inaccuracy on the results, the above procedure is carried out 10 times with the reference snooker ball placed in different positions. The X and Y co-ordinates of points *P1*, *P2*, and the resulting values of X_C and Y_C are tabulated in table 4.8, from which the average values of X_C and Y_C are computed as -3.238 mm and -53.173 mm respectively. It can therefore be concluded that the tool definition of the on-board camera 'CAMTOOL' equals (-3.238, -53.173, 0, 90, -90, 0), as illustrated in figure 4.31.

4.3 Conclusions

The process of accurately determining the position of snooker balls on the snooker table by machine vision is covered in this chapter. Two CCD cameras are incorporated into the system, one for providing an overall view of the snooker table and the other to focus at a single ball. Because of the machine vision hardware, images are digitised in black and white with 64 gray-scale levels. Contemporary machine vision systems operate with 256 gray-scale levels. Despite this hardware deficiency, reliable identification of snooker ball colours is achieved through carefully controlled scene illumination and analysis of the gray-scale maps of the various colours.

The overhead camera covers a relatively large area (the entire snooker table measures 1.56 meter by 0.84 meter) in world space at the expense of resolution. Hence positional data of snooker balls obtained through the overhead camera are only accurate to within 10 mm. The on-board camera, on the other hand, is used to focus on a single ball and produces images of much higher resolution. By first finding the approximate position of a ball through the overhead camera and then applying visual servoing with the on-board camera, the position of a ball can be determined to within 0.32 mm and 0.19 mm in the X and Y direction respectively.

An original method of robot hand-to-eye calibration is also presented. The simplicity and effectiveness of the calibration process is a consequence of the a priori knowledge of object characteristics and scene orientation. After correct calibration of the camera, it behaves as a 7^{th} link of the PUMA robot arm and provides visual sensory data essential for the functioning of the servoing mechanism.

It would be expensive in terms of execution time to perform visual servoing on every snooker ball on the table. It could take up to 10 minutes to complete if all 17 snooker balls are on the table. Consequently it is decided to perform visual servoing only on the two balls involved in a shot i.e. the cue ball and the target ball. Together with the position of the pocket which is predetermined, the shot angle can be computed by elementary geometry. The disadvantage of this compromise is that the shot planning functions use ball positional data derived from the overhead camera that are accurate to within 10 mm only. Fortunately, the system can be easily modified to perform visual servoing on every ball when, for example, the total number of balls on the table is 10 or less should the need arises. For the present set-up, no such strategy is implemented.

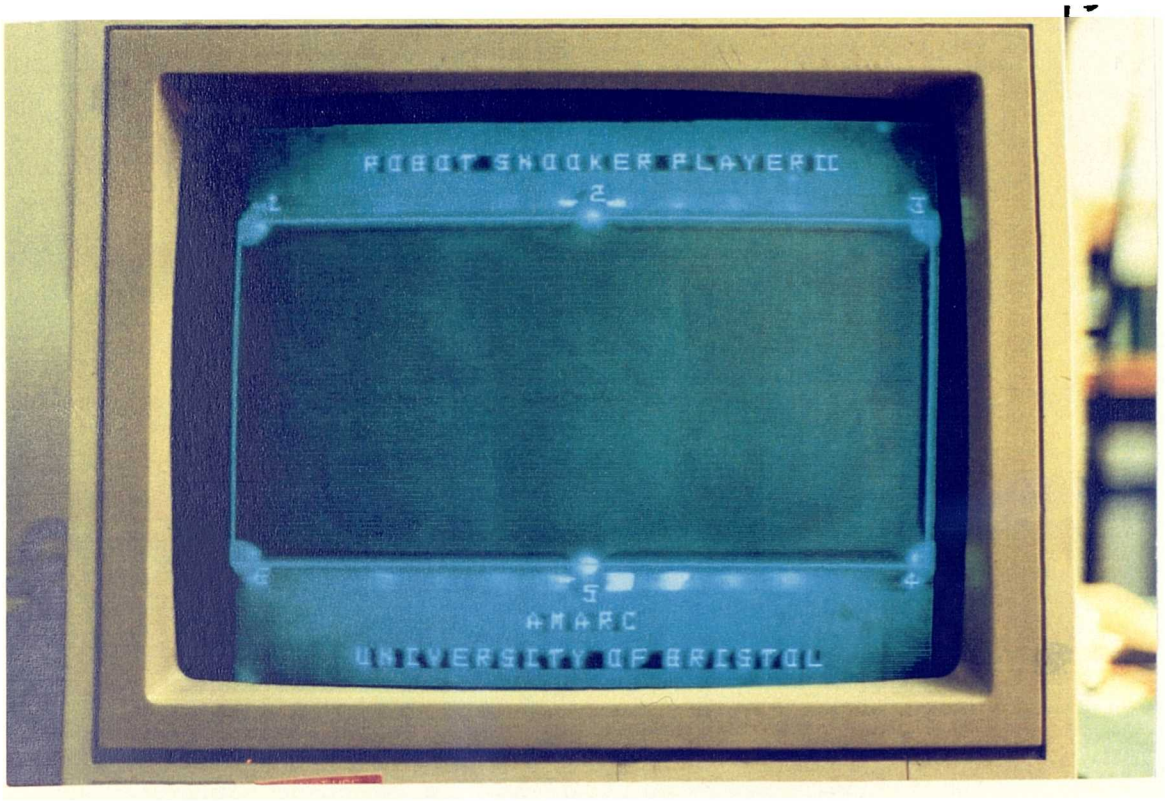
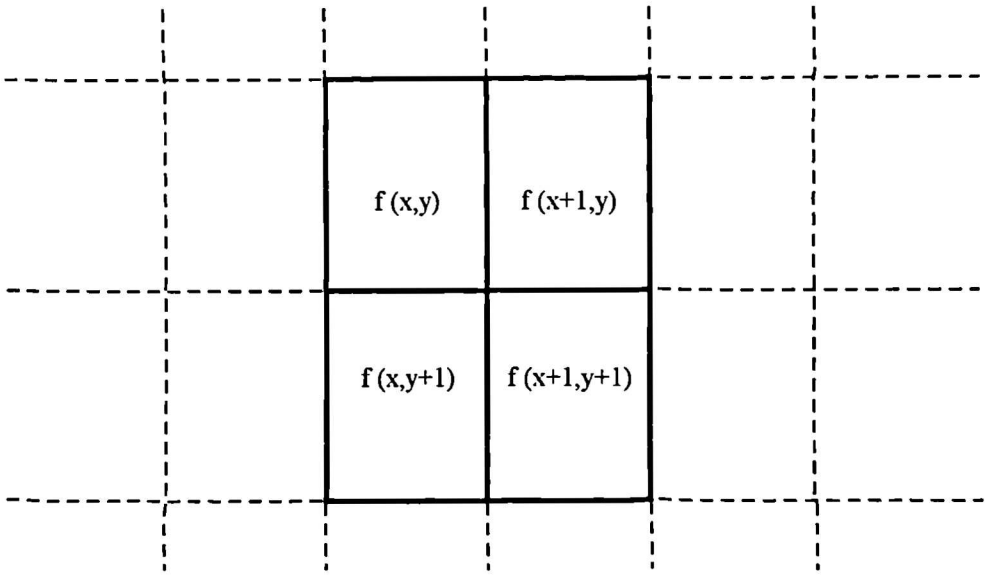


Figure 4.1 Digitised Image of Snooker Table



$$G[f(x,y)] = |f(x,y) - f(x+1,y+1)| + |f(x,y+1) - f(x+1,y)|$$

where $f(x,y)$ = gray-scale value of pixel at (x,y) in pixel coordinates

$G[f(x,y)]$ = gray-scale value of ENHANCED pixel at (x,y)

Figure 4.2 Roberts Gradient Filter

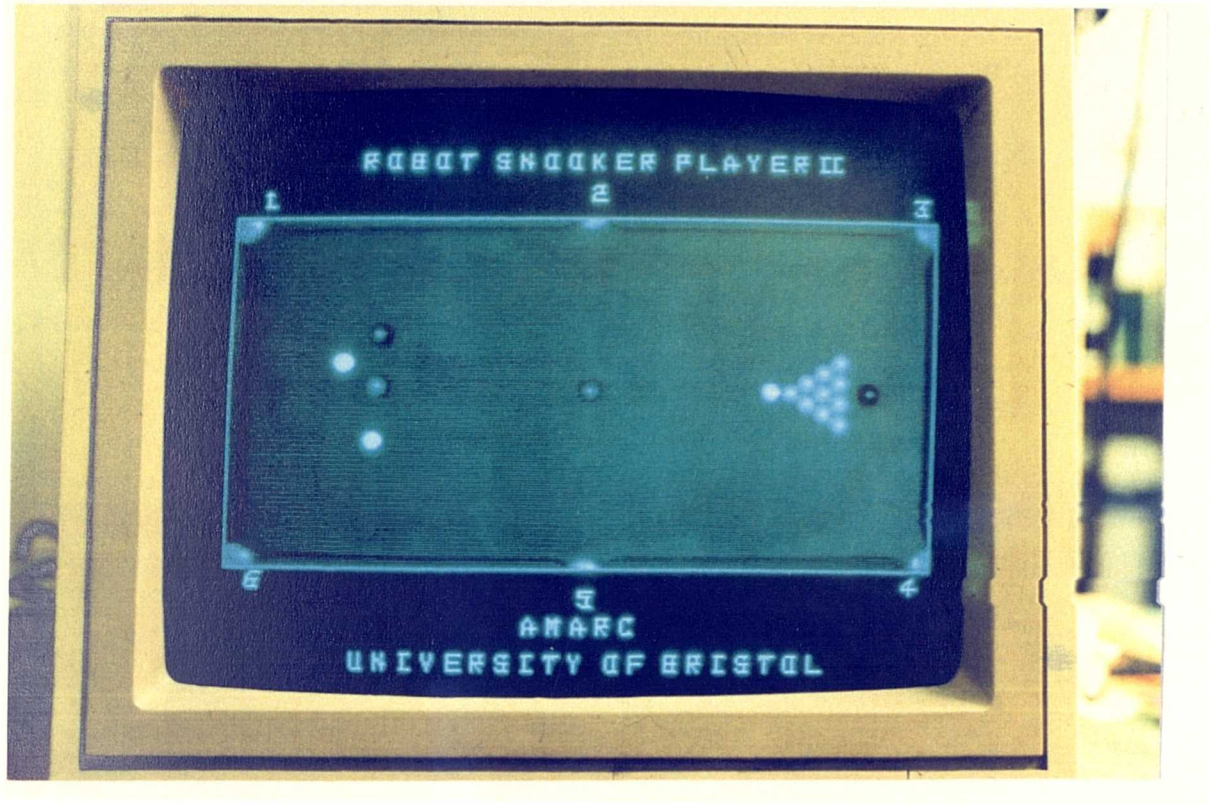


Figure 4.3(a) Image of Snooker Balls Before Sharpening

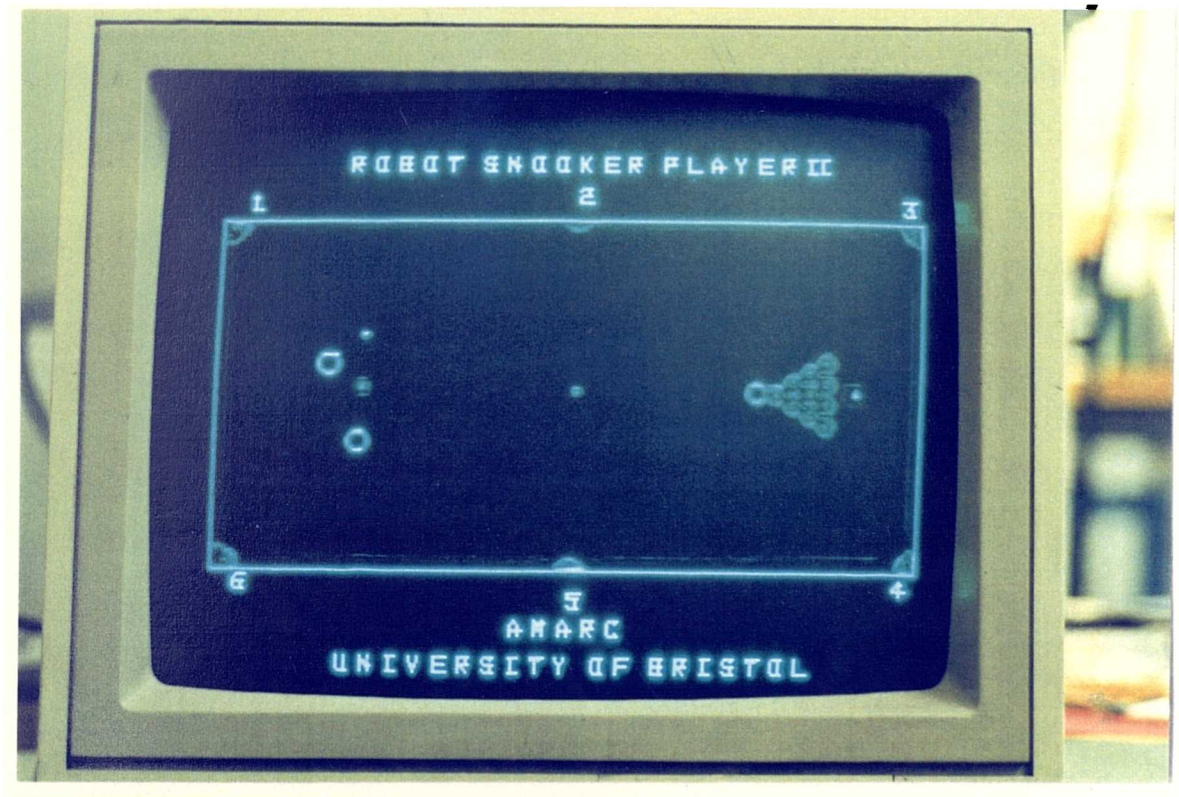


Figure 4.3(b) Image After Sharpening Using Roberts Gradient

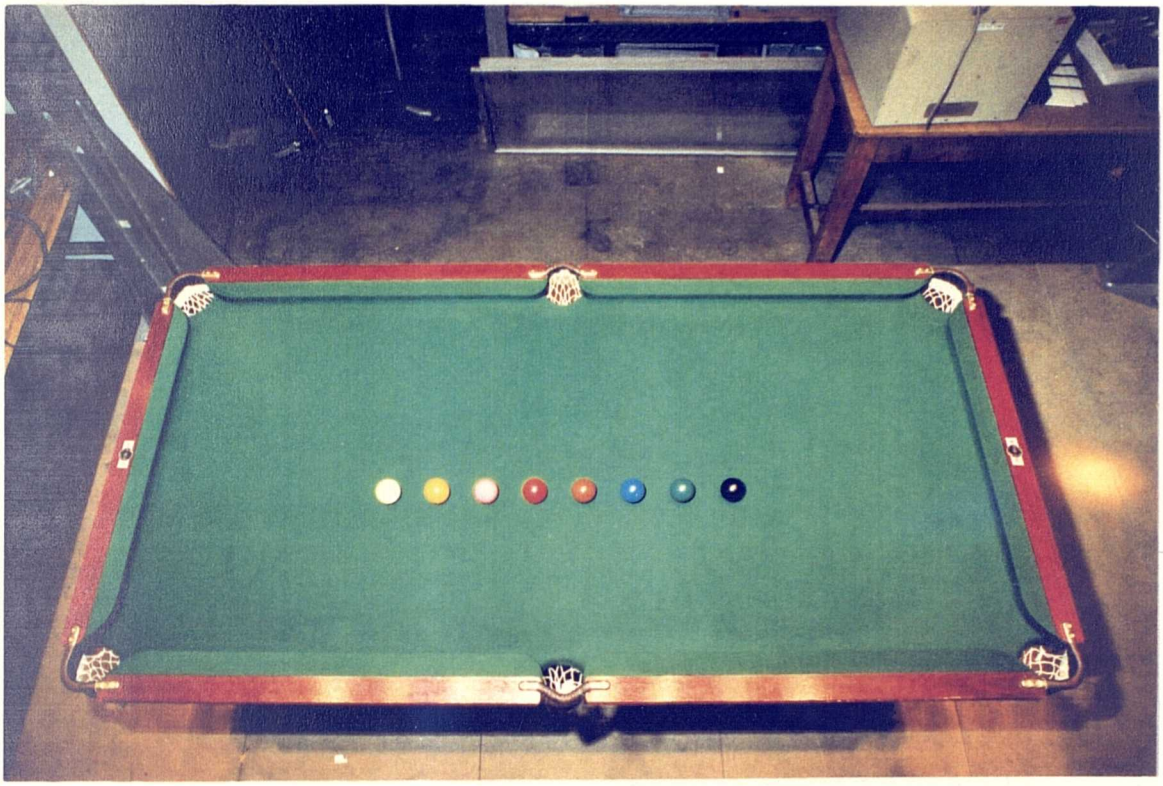


Figure 4.4 8 Colour Balls on The Snooker Table Viewed in Colour

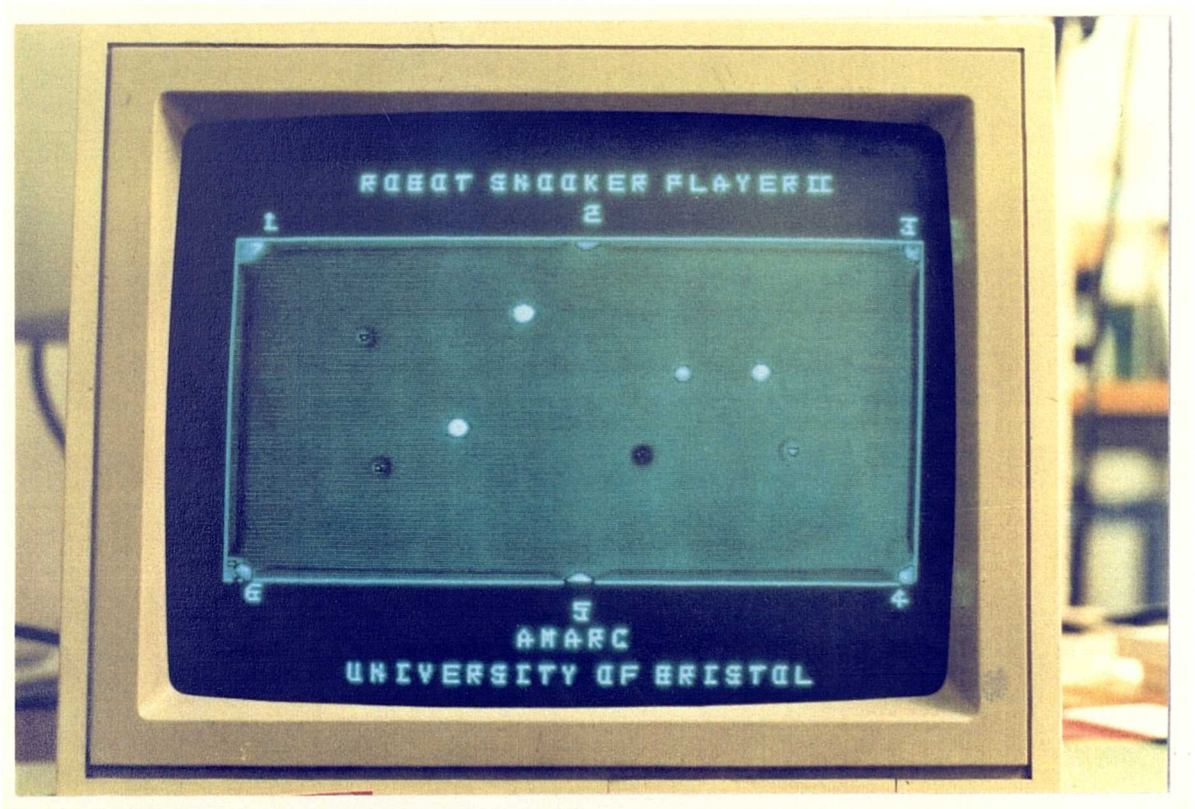


Figure 4.5 Gray-Scale Segmentation of the 8 Colour Balls

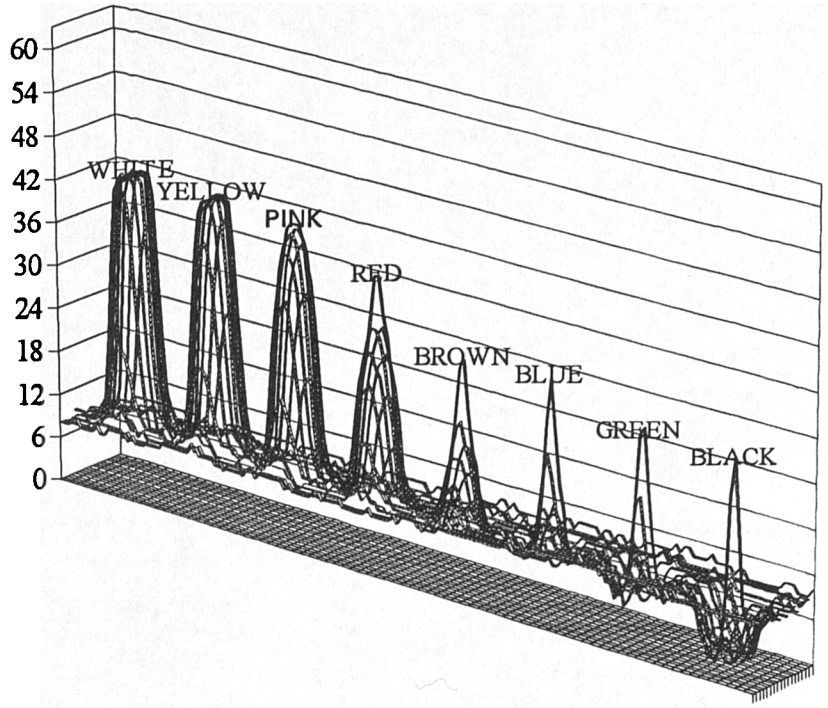


Figure 4.6 3-D Gray-Scale Profile of Snooker Balls

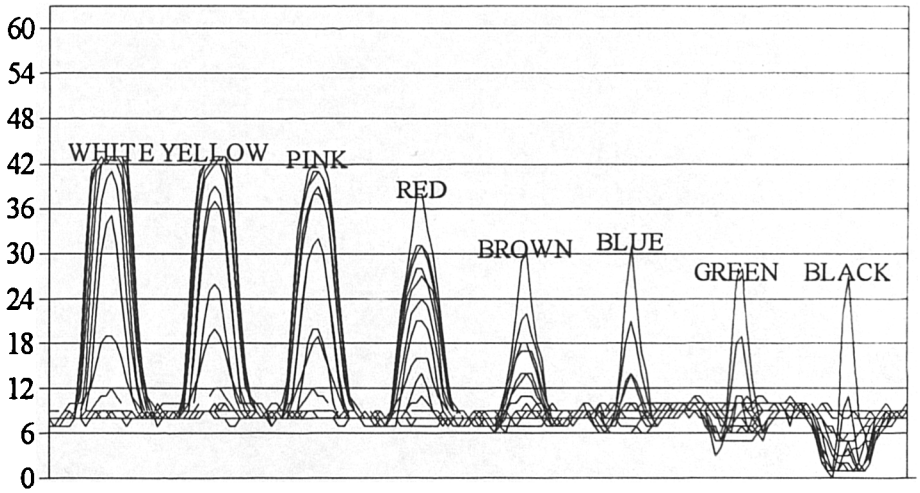


Figure 4.7 2-D Gray-Scale Profile of Snooker Balls

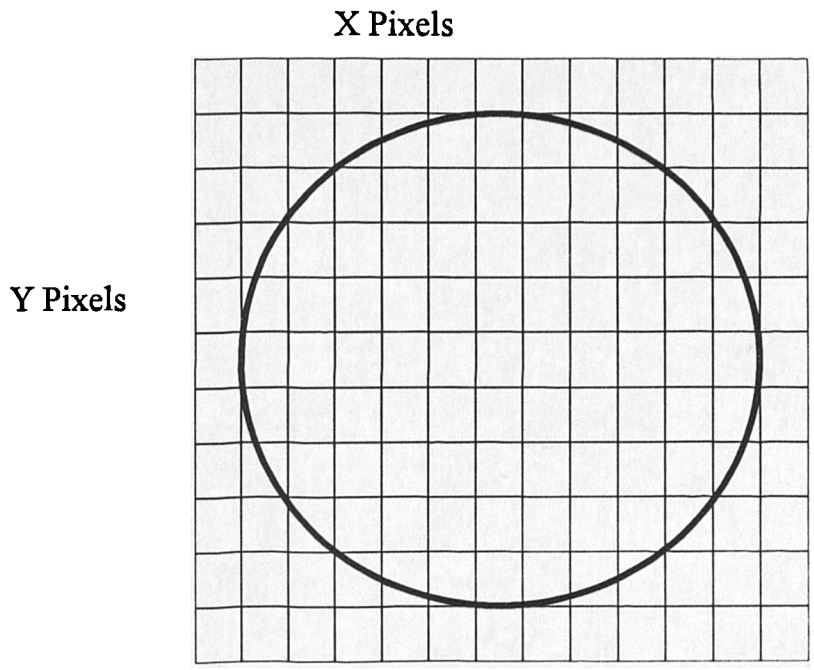


Figure 4.8 Area Covered by a Snooker Ball in Pixel Map

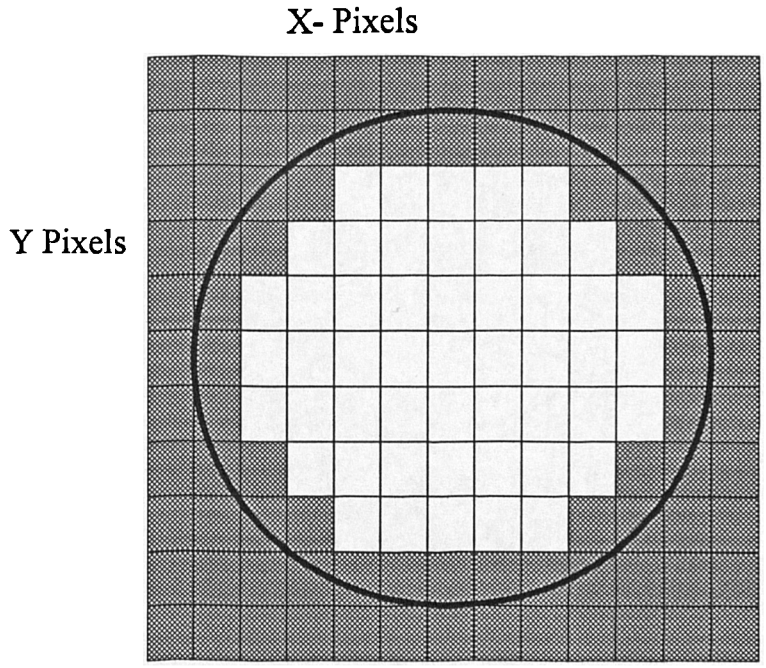


Figure 4.9 (a) Pixels Lying Outside or On The Edge of a Ball

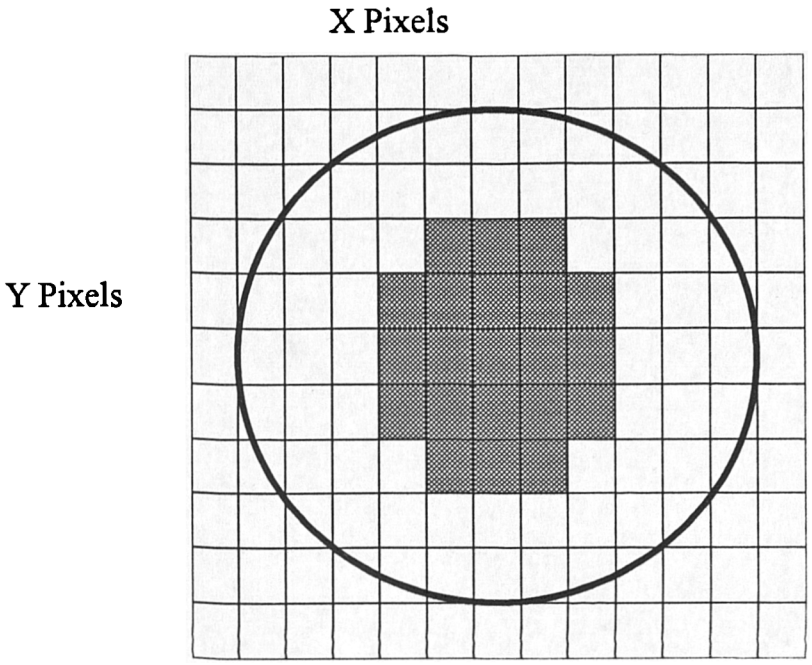
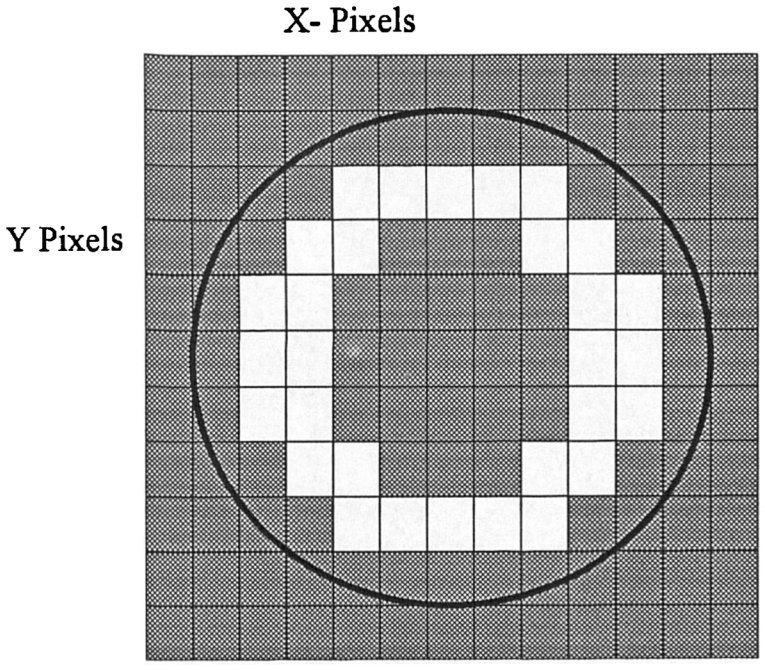


Figure 4.9 (b) Pixels Containing Lighting Reflection



Values of unshaded pixels are used in computing the average gray-scale value representing a colour.

Figure 4.9 (c) Average Gray-Scale Mask

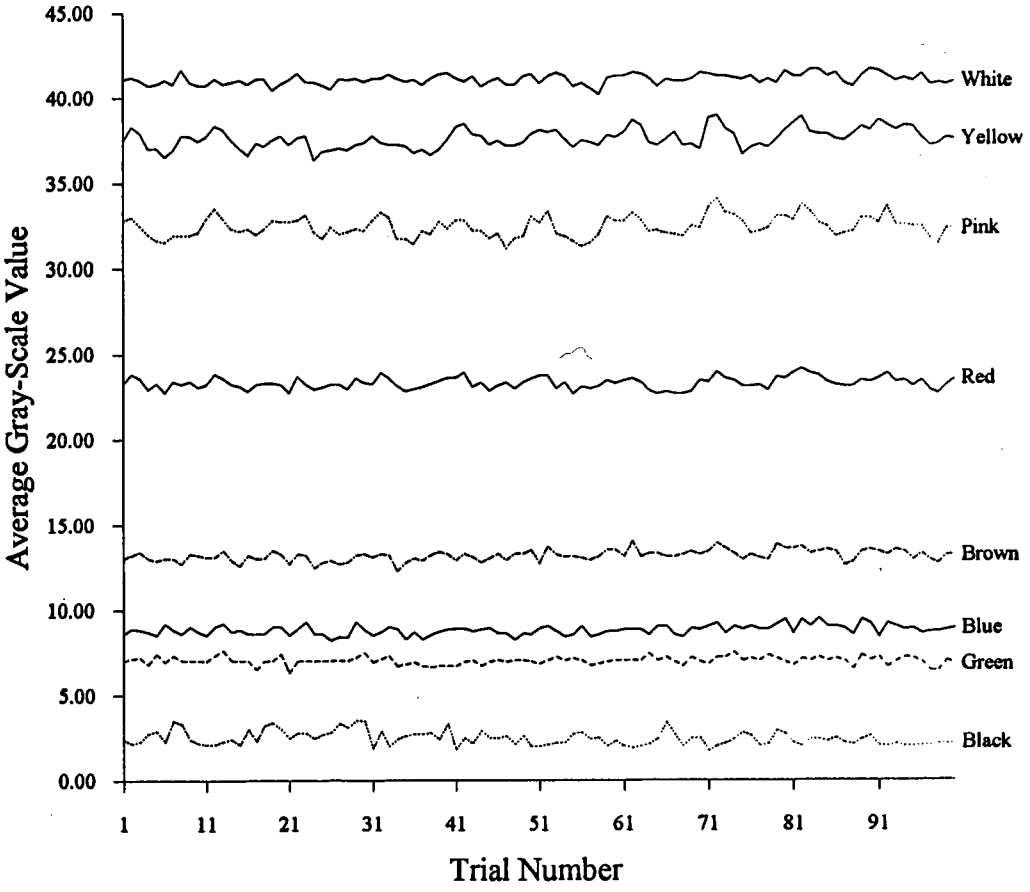


Figure 4.10 Gray-Scale Separation of Ball Colours

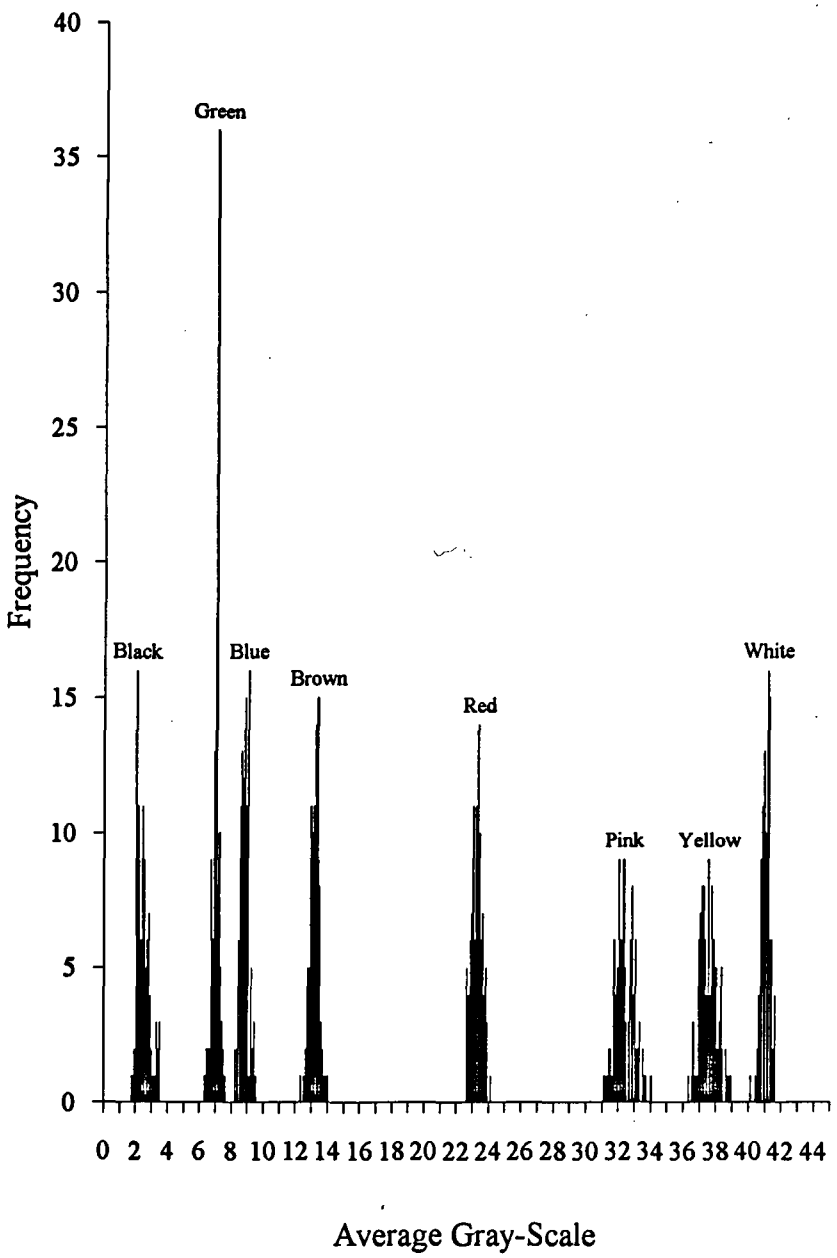


Figure 4.11 Frequency Histogram of Average Gray-Scale Values of Snooker Ball Colours

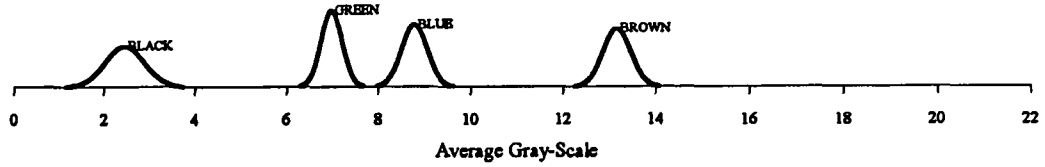


Figure 4.12 Average Gray-Scale Distribution of Dark Colour Balls

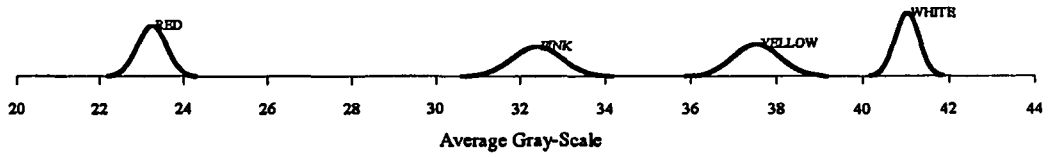


Figure 4.13 Average Gray-Scale Distribution of Bright Colour Balls

Constants
red = 1
yellow = 2
green = 3
brown = 4
blue = 5
pink = 6
black = 7
white = 8

Figure 4.14 Constant Declaration

	j=1	2
i=1	x1	y1
2	x2	y2
3	x3	y3
4	x4	y4
5	x5	y5
6	x6	y6
7	x7	y7
8	x8	y8
9	x9	y9
10	x10	y10

REDPOS [i,1] = x co-ordinate of red ball no. i
REDPOS [i,2] = y co-ordinate of red ball no. i
i = 1..10

Figure 4.15 Array ' REDPOS '

	j=1	2	
i=1			Not Used
2	x2	y2	
3	x3	y3	
4	x4	y4	
5	x5	y5	
6	x6	y6	
7	x7	y7	
8	x8	y8	

COLOURPOS [c,1] = x co-ordinate of colour ball c
COLOURPOS [c,2] = y co-ordinate of colour ball c
c = {yellow, green, brown, blue, pink, black, white}

Figure 4.16 Colour Constants and Array ' COLOURPOS '

Figure 4.17 A Sample Pack of 5 Red Balls

```
FOR every pack DO
BEGIN
        THRESHOLD = 19
        Set current window to that associated with this pack
        REPEAT
                THRESHOLD = THRESHOLD + 1
                PICTURE
                FOR any object that is NOT a pack
                        identify its colour and store its position
        UNTIL no object inside window is a pack
```

Figure 4.18 Pseudo-code For Resolving Packs of Snooker Balls

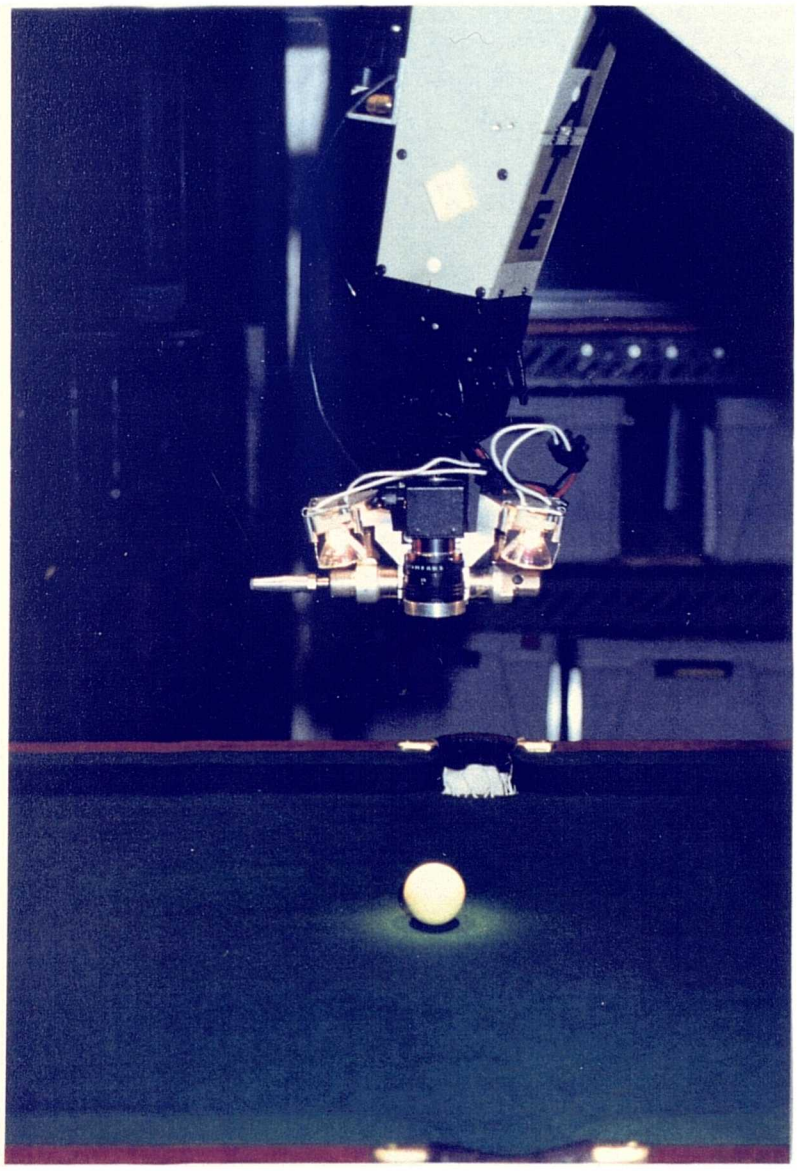


Figure 4.19 On-board Camera and Lighting Assembly Fitted to PUMA Arm

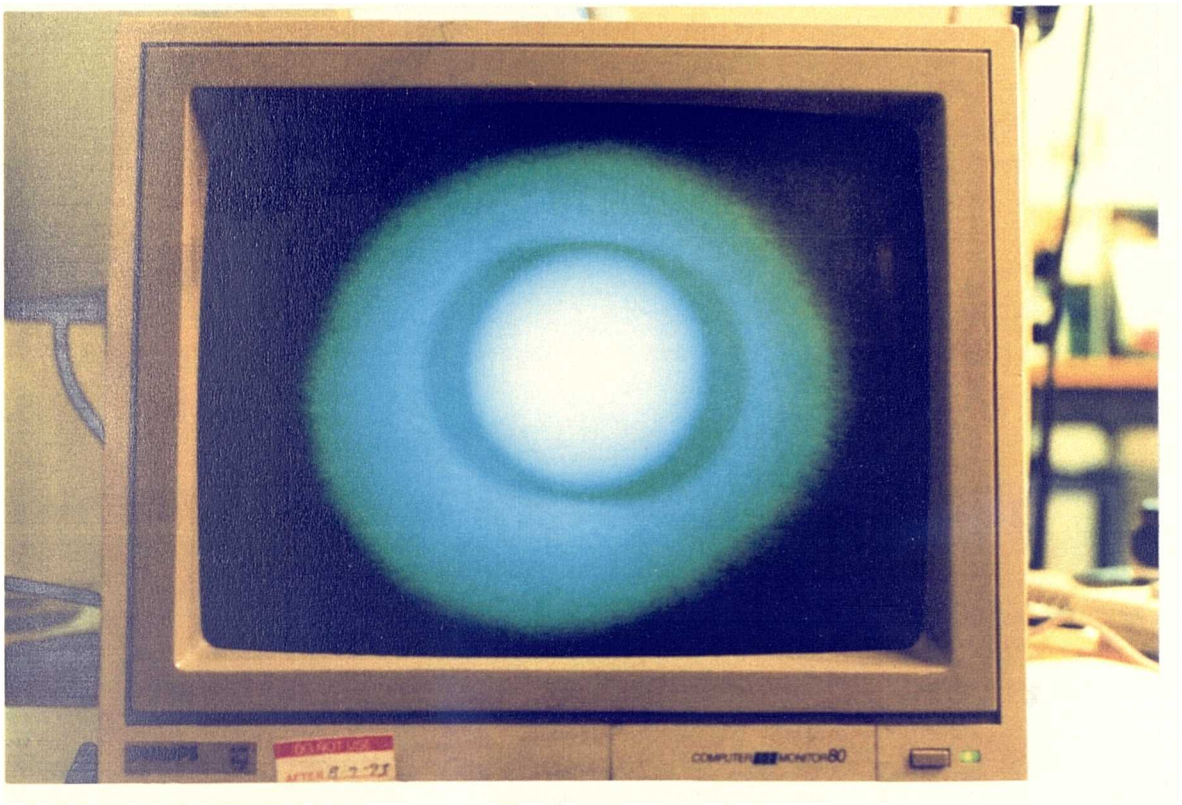


Figure 4.20 On-board Camera View of a Snooker Ball

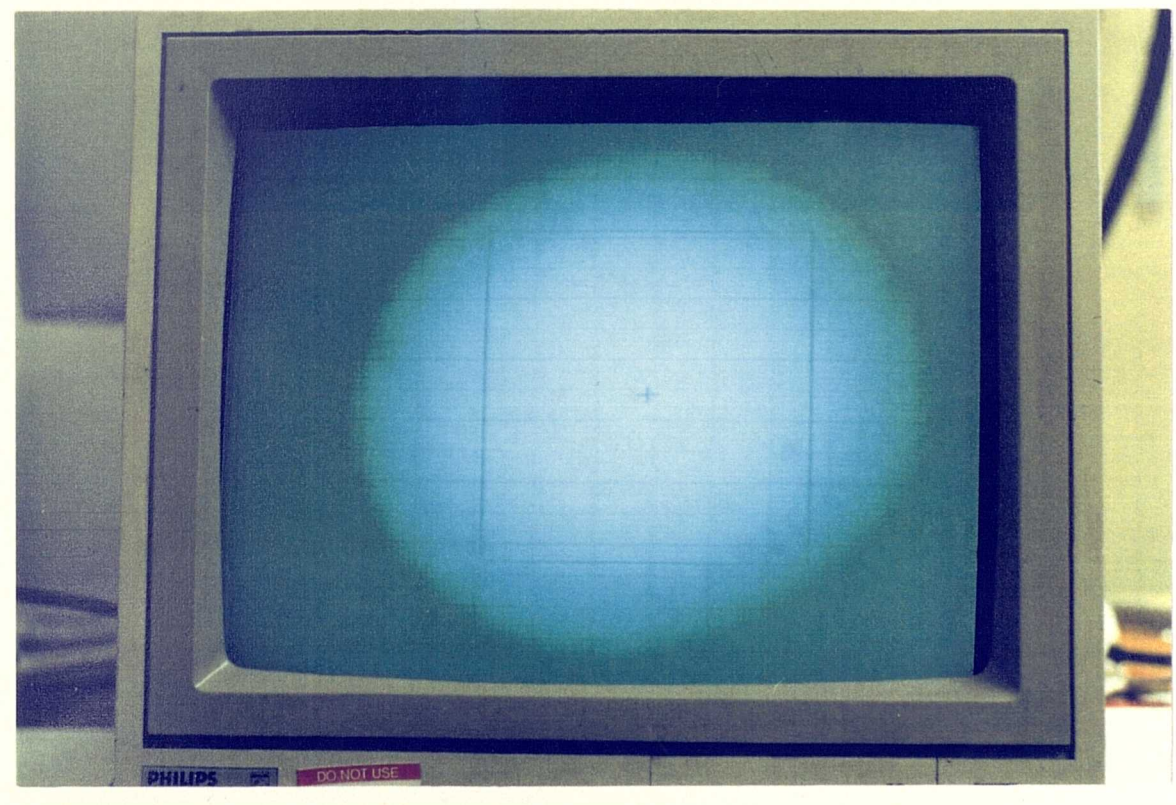


Figure 4.21 Correspondence Between On-board Camera View Window in Pixel and Millimeter Scale

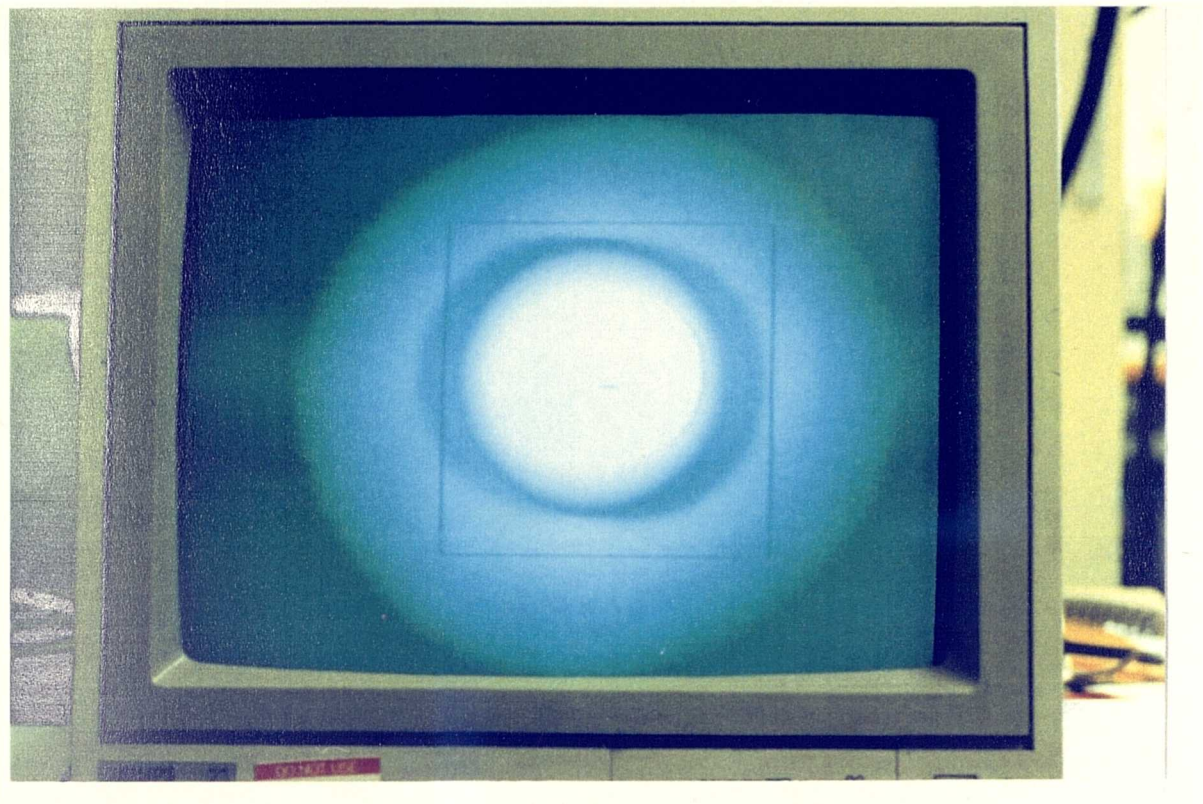
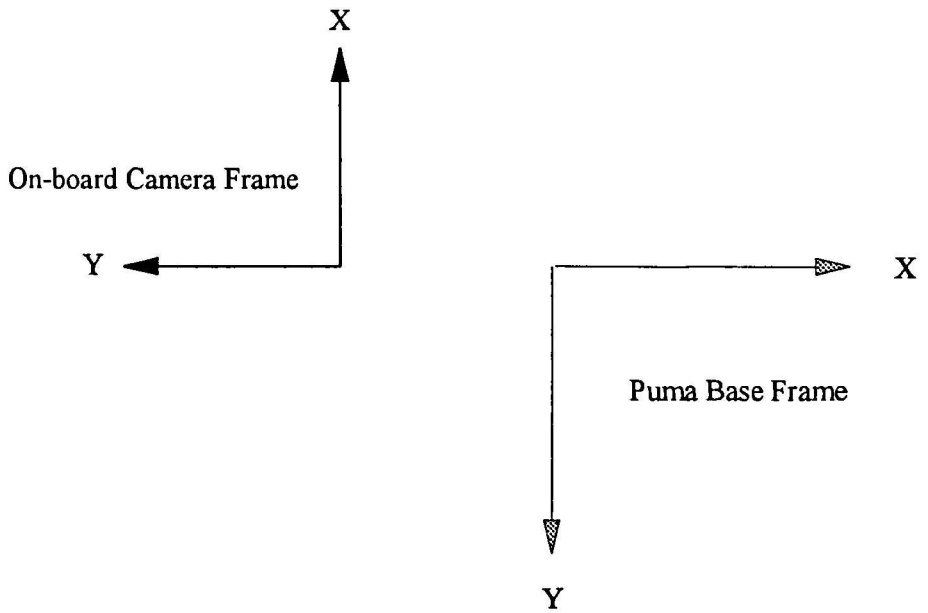
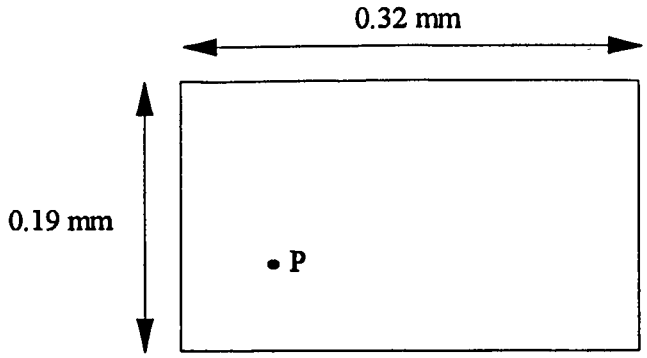


Figure 4.22 Sample Image of a Snooker Ball Inside the On-board Camera View Window



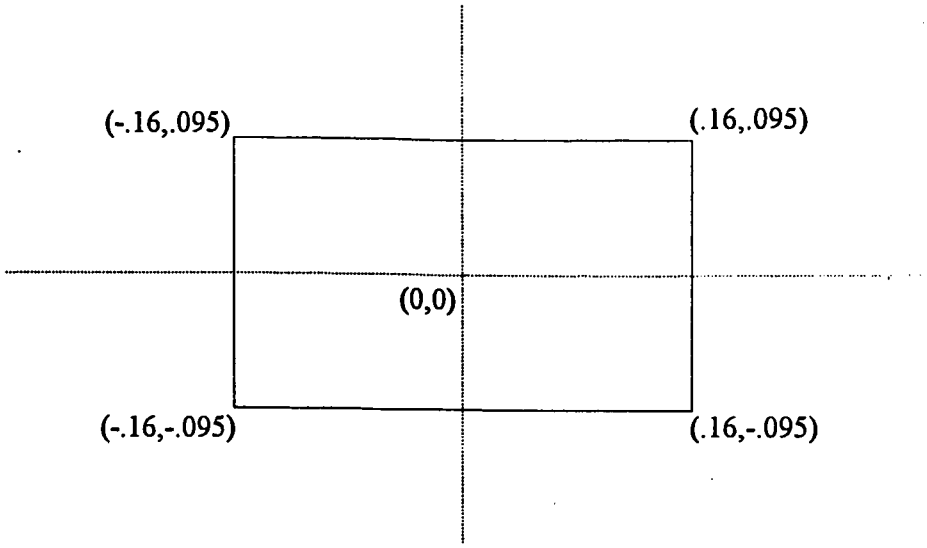
Note : 1. This is a plan view from above the snooker table
2. All axes shown are positive.
3. The positive Z-axis is perpendicular to the paper away going away from the viewer
4. The world frame has the same orientation as the Puma base frame.

Figure 4.23 Relationship Between On-board Camera and Puma Coordinate Frames



Rectangle K in Image Space

(a) Image Area Mapping Onto 1 On-Board Camera Pixel



Any point (x,y) where |x| <= .16 and |y| <= .095
corresponds
to the same pixel on the CCD through the lens

(b) Cartesian Coordinate System Superimposed onto Image Area

Figure 4.24 Resolution Approximation

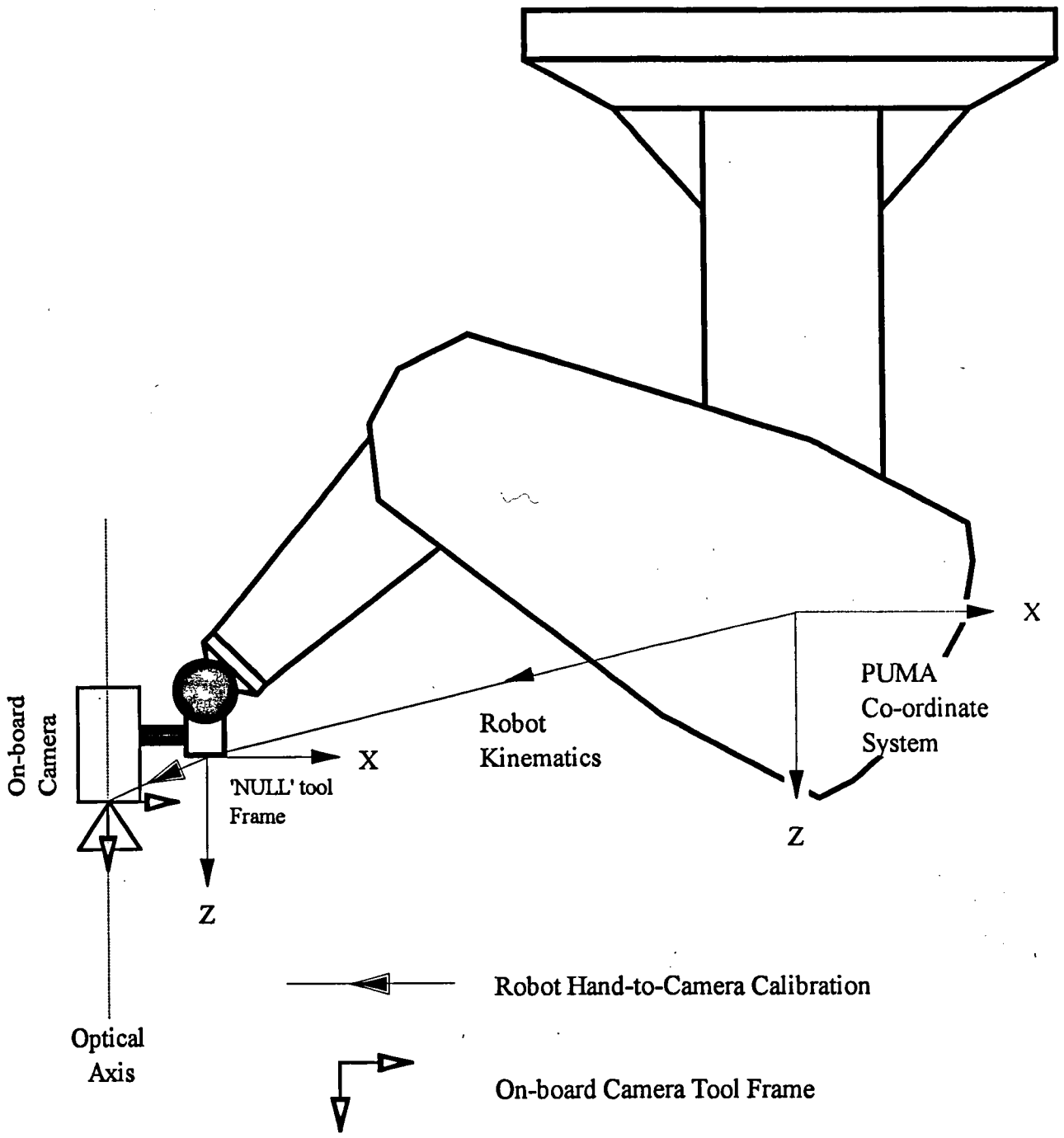


Figure 4.25 Robot Hand-to Eye Calibration

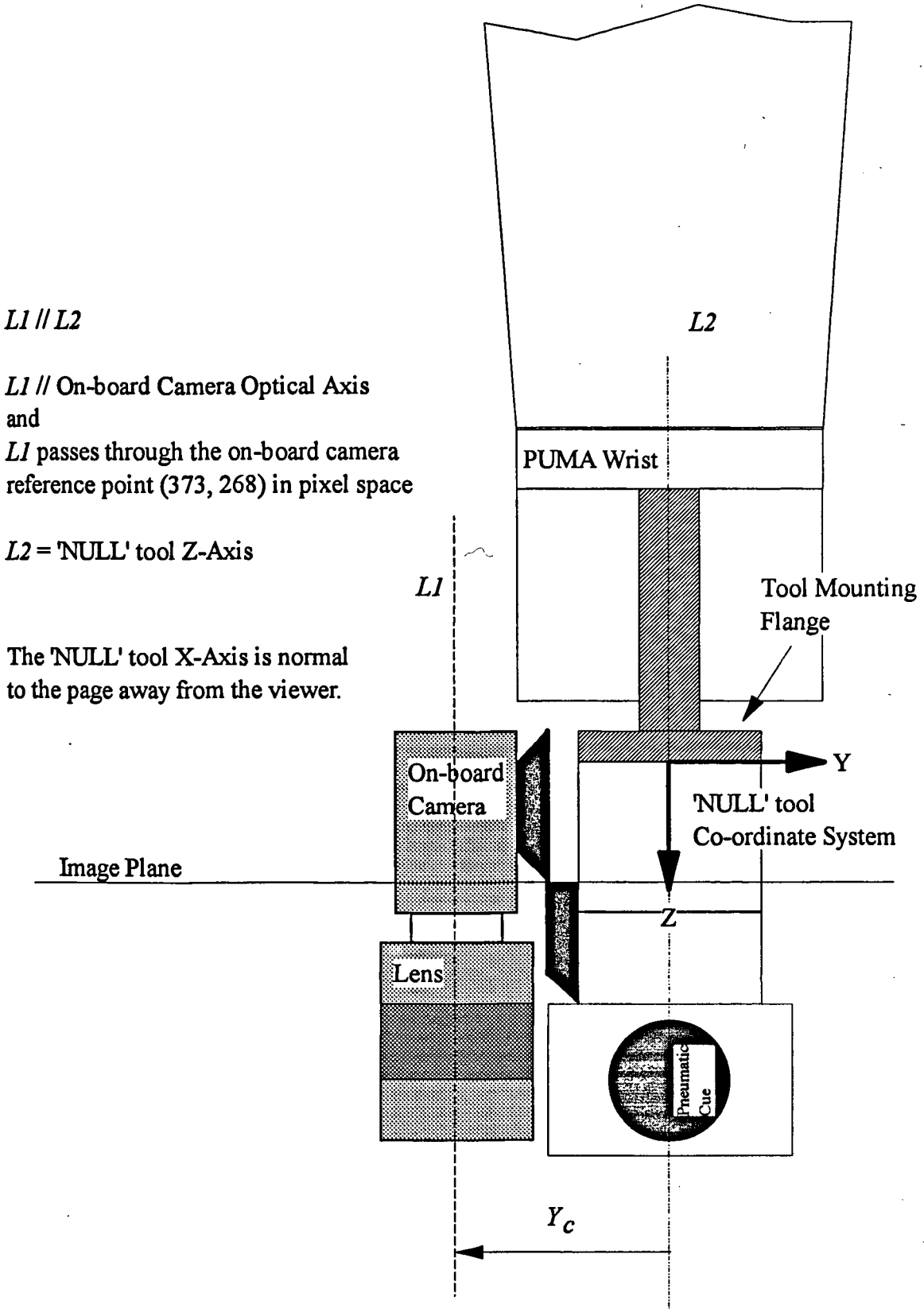


Figure 4.26 (a) Position of On-board Camera Relative to PUMA 'NULL' Tool

L1 // *L2*

L1 // On-board Camera Optical Axis
and
L1 passes through the on-board camera reference point (373, 268) in pixel space

L2 = 'NULL' tool Z-Axis

The 'NULL' tool Y-Axis is normal to the page away from the viewer.

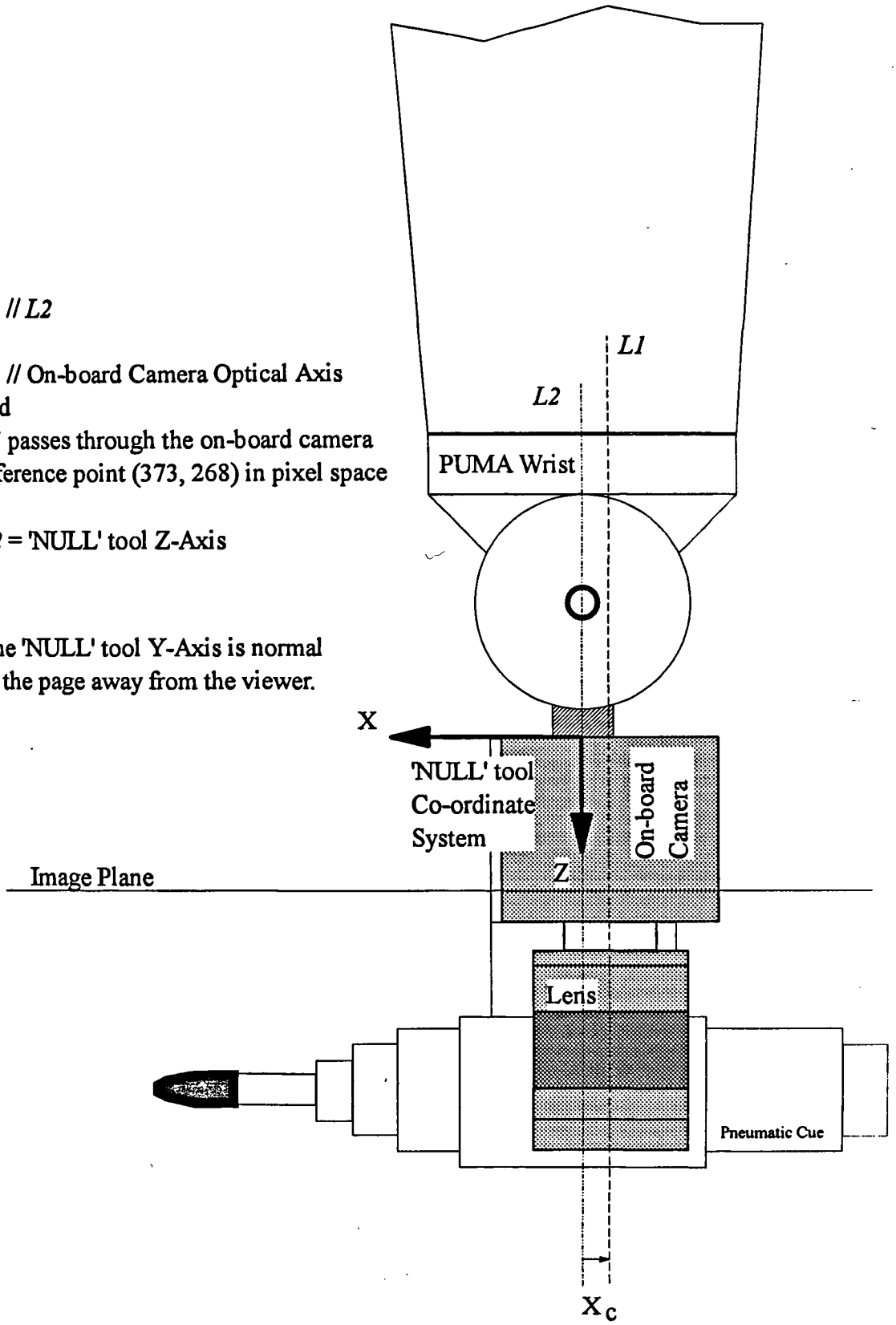
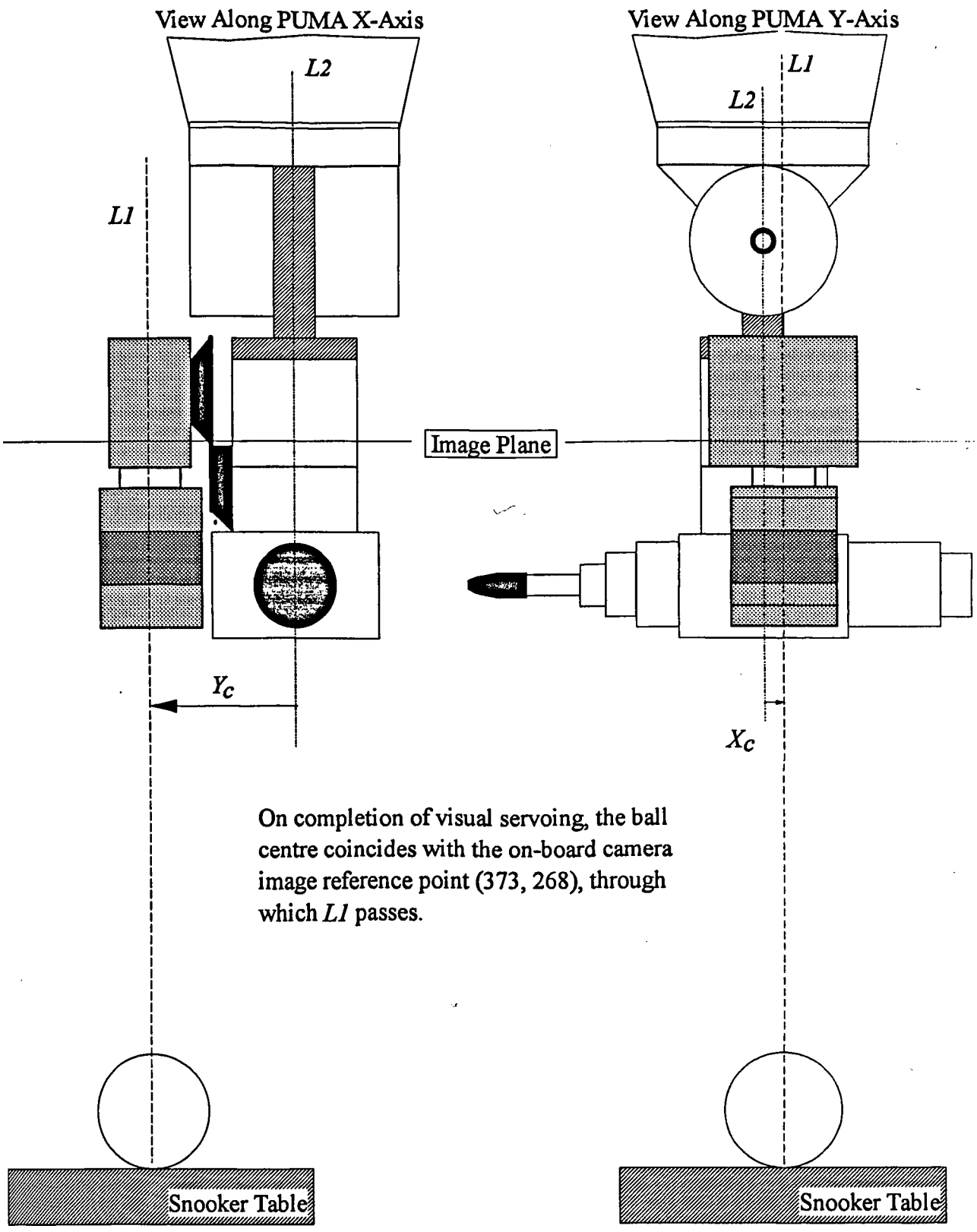
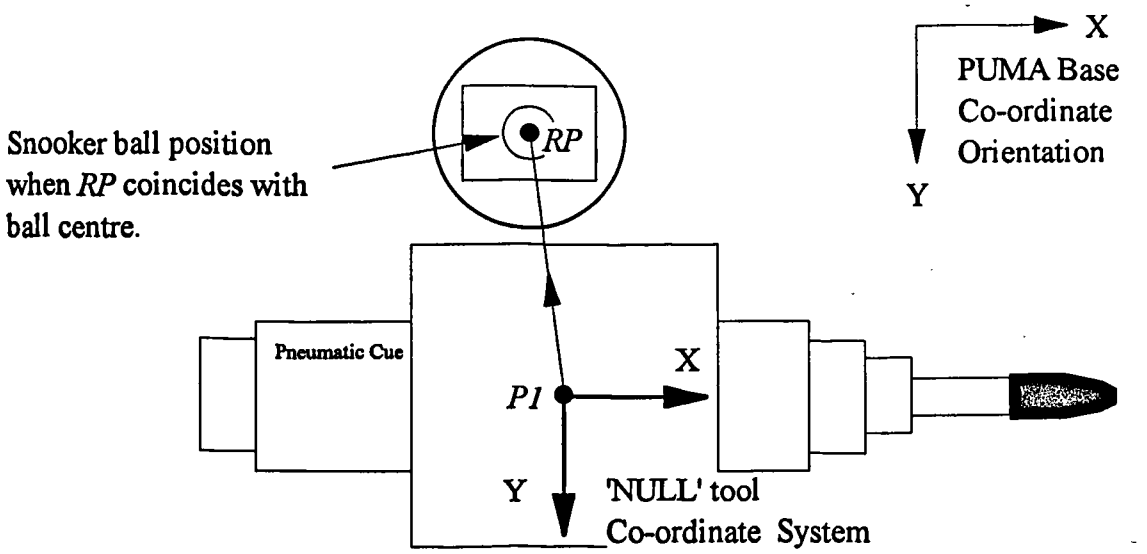


Figure 4.26 (b) Position of On-board Camera Relative to PUMA 'NULL' Tool



Note : This figure shows the PUMA robot arm, fixed at a particular position and orientation, viewed from 2 angles.

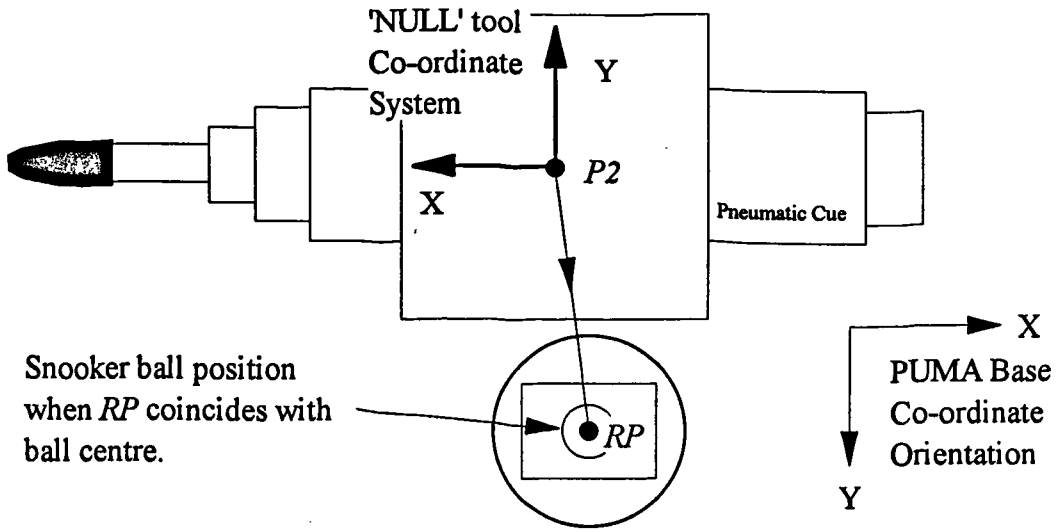
Figure 4.27 Two Views of a Snooker Ball After Visual Servoing is Completed



$P1 = (x1, y1, 276, 90, -90, 0)$

The Z-axes of both the PUMA base and 'NULL' tool co-ordinates are normal to the page away from the viewer.

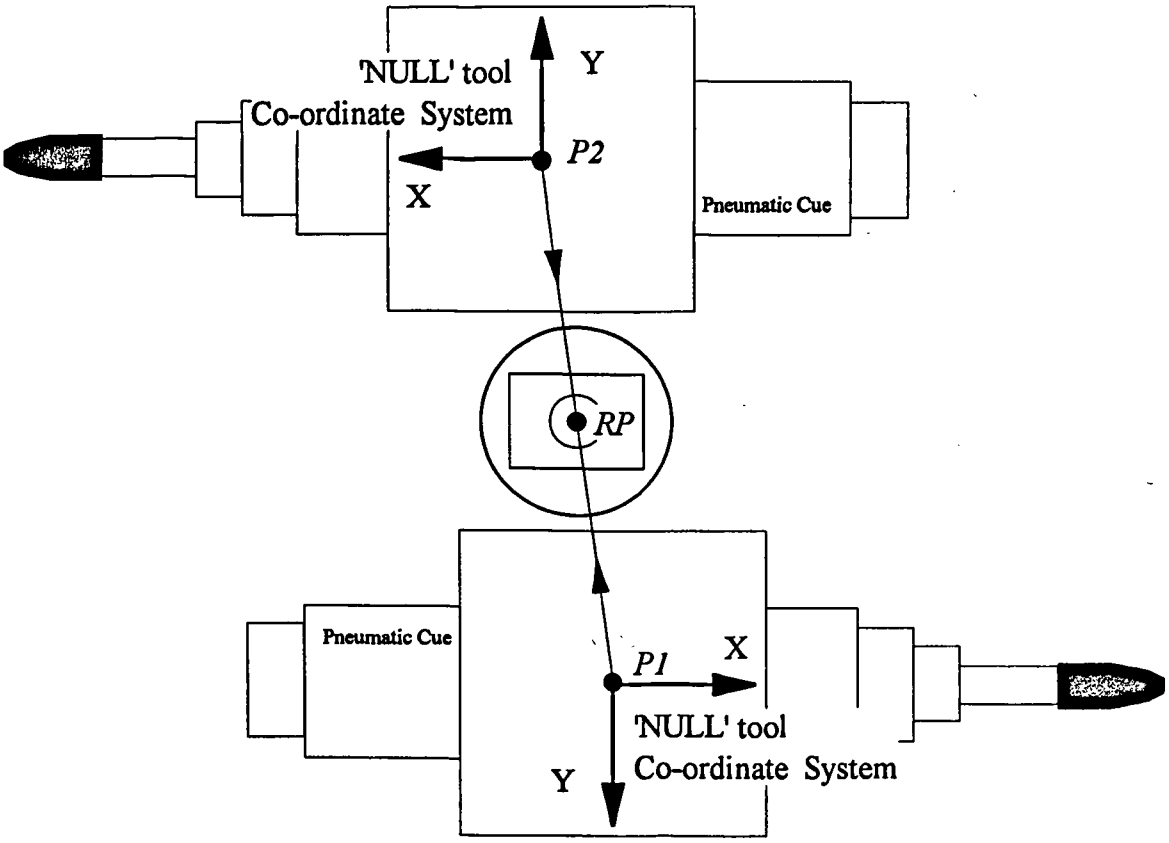
Figure 4.28 Overlapping *RP* with Ball Centre with 'NULL' Tool in (90, -90, 0) Orientation



$P2 = (x2, y2, 276, -90, -90, 0)$

The Z-axes of both the PUMA base and 'NULL' tool co-ordinates are normal to the page away from the viewer.

Figure 4.29 Overlapping *RP* with Ball Centre with 'NULL' Tool in (-90, -90, 0) Orientation



Given :

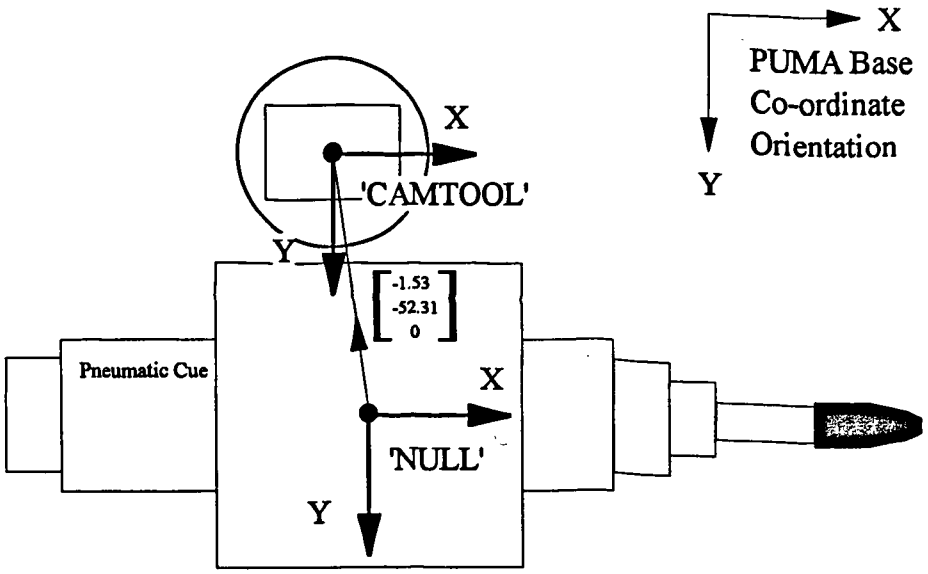
1. $|P1RP| = |P2RP|$
2. $\angle P1RPP2 = 180°$
3. $P1 = (x1, y1, 276, 90, -90, 0)$
4. $P2 = (x2, y2, 276, -90, -90, 0)$

It can be deduced that :

$RP = (Xc, Yc, 0)$ where $Xc = (x2-x1)/2$ and $Yc = (y2-y1)/2$ in 'NULL' tool co-ordinates

Note : Since the optical axis and 'NULL' tool Z-axis are parallel, the Z co-ordinate of *RP* can be arbitrarily set to zero.

Figure 4.30 Computation of On-board Camera Reference Point Position in 'NULL' Tool Co-ordinates



'NULL' Tool Definition = (0, 0, 0, 90, -90, 0)

'CAMTOOL' in 'NULL' Tool Co-ordinates = (-1.53, -52.31, 0)

'CAMTOOL' Orientation = 'NULL' Tool Orientation = (90, -90, 0)

=> 'CAMTOOL' Definition = (-1.53, -52.31, 0, 90, -90, 0)

Note : All Z-Axes are normal to the page away from the viewer.

Figure 4.31 'CAMTOOL' Definition

y-pixel \ x-pixel	WHITE ball centre = (94,138)																			
	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104
130	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	8	9	9	8	9
131	7	7	8	8	7	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
132	8	8	8	8	8	8	8	9	9	8	8	8	8	8	8	8	8	8	8	8
133	8	8	8	8	8	9	9	9	9	9	8	8	7	8	8	8	8	8	8	8
134	8	8	8	8	9	13	22	30	34	35	31	24	18	11	9	9	8	8	8	9
135	7	7	8	8	14	27	37	42	42	42	42	40	30	20	13	10	8	9	7	8
136	8	9	9	12	25	39	42	43	42	43	43	43	40	29	19	13	11	10	9	9
137	8	8	8	16	30	41	42	42	43	43	43	43	43	34	22	14	10	8	9	8
138	8	8	9	17	31	41	42	42	43	43	43	43	43	35	23	15	11	9	9	8
139	8	8	8	13	28	40	42	43	42	43	43	43	42	33	20	14	11	9	9	9
140	9	9	9	11	20	32	41	42	42	43	43	42	36	25	17	12	9	8	10	9
141	8	8	9	9	11	19	28	37	40	41	39	33	24	17	12	9	9	9	9	8
142	9	10	10	10	11	12	15	18	19	19	18	16	13	11	10	9	9	9	9	10
143	8	8	8	8	8	9	10	11	11	12	11	10	10	9	8	9	8	8	8	8
144	9	9	9	9	9	10	10	10	11	10	10	10	10	9	9	9	9	9	9	9
145	8	8	8	8	8	9	9	9	9	9	9	9	8	9	8	8	8	8	8	8
146	9	9	8	8	8	8	9	8	8	9	9	9	9	9	8	9	8	8	8	9

Table 4.1 (a) Gray-Scale Map of WHITE Ball

y-pixel \ x-pixel	YELLOW centre = (114,138)																			
	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124
130	8	8	8	8	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
131	8	8	8	8	8	8	8	9	9	8	8	8	8	8	8	8	8	8	8	8
132	9	9	8	8	8	8	9	9	9	9	9	9	9	9	8	8	8	8	8	9
133	8	8	8	8	8	8	8	9	9	9	8	8	7	7	8	8	8	9	8	8
134	8	9	8	9	9	10	15	21	25	26	25	20	15	11	9	8	9	9	8	8
135	8	8	8	8	9	18	27	34	38	39	38	34	26	18	11	9	8	8	8	8
136	9	9	9	10	18	29	37	41	42	42	42	42	36	27	19	13	10	10	10	9
137	8	8	8	10	22	34	40	42	42	43	43	43	40	32	21	13	10	9	8	8
138	9	9	8	12	23	35	41	42	42	43	43	43	42	33	23	15	11	9	9	9
139	9	9	9	11	21	33	40	42	42	42	43	43	40	32	22	14	10	9	9	8
140	8	9	9	9	16	26	35	40	41	42	42	40	34	26	18	12	11	10	9	9
141	8	8	8	9	10	16	25	32	36	37	36	32	25	18	12	10	9	9	9	9
142	10	10	10	10	10	12	14	17	19	20	19	17	14	12	10	10	10	10	9	9
143	8	8	8	8	9	9	10	11	11	12	12	11	10	9	8	8	8	8	8	9
144	9	9	9	9	9	10	10	11	11	11	11	10	10	10	9	9	9	9	9	9
145	8	8	9	9	9	9	9	9	9	10	9	10	9	9	9	8	8	8	8	8
146	9	9	9	9	9	9	9	9	9	9	10	10	9	9	9	9	9	9	8	9

Table 4.1 (b) Gray-Scale Map of YELLOW Ball

x-pixel PINK centre = (134,138)

y-pixel	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
130	9	9	8	9	9	9	9	9	9	9	9	9	9	9	8	9	9	9	9	8
131	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
132	8	8	8	8	9	9	8	9	9	8	8	8	8	9	8	8	8	8	8	8
133	9	8	8	9	8	8	8	8	8	8	8	8	8	7	7	8	8	8	8	8
134	8	8	8	8	8	9	13	18	20	20	18	15	10	8	8	8	8	8	8	8
135	7	8	8	8	10	17	24	29	31	32	30	25	19	13	9	8	8	8	7	7
136	9	9	8	10	17	26	33	36	38	39	37	34	28	20	13	10	9	9	8	8
137	8	8	8	11	21	30	36	39	41	41	40	38	32	23	16	11	9	8	8	8
138	8	9	10	13	23	33	37	41	42	42	42	40	34	25	17	11	10	9	9	9
139	9	8	8	12	21	30	36	39	40	41	40	37	32	24	16	11	9	8	9	8
140	9	9	9	10	17	25	31	35	38	38	37	34	27	20	14	11	9	9	9	9
141	9	9	8	9	11	17	23	29	31	32	30	25	20	14	10	8	9	8	8	8
142	10	10	9	10	10	12	14	16	18	19	17	15	13	11	9	9	9	9	9	9
143	7	8	8	8	9	9	10	11	12	11	11	10	9	8	8	8	8	8	8	8
144	9	9	9	9	9	9	9	10	10	10	10	10	9	9	8	8	9	9	9	9
145	8	8	8	9	9	8	9	9	9	9	9	9	9	8	8	8	8	8	8	8
146	9	9	9	9	9	9	9	9	9	9	9	9	9	8	9	8	8	8	8	8

Table 4.1 (c) Gray-Scale Map of PINK Ball

x-pixel RED centre = (154,138)

y-pixel	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164
130	8	9	8	8	9	9	9	9	9	9	9	9	9	9	8	8	9	9	8	9
131	8	8	7	8	8	8	8	8	8	8	8	8	8	8	8	8	7	7	8	8
132	8	8	8	8	8	8	8	7	8	8	8	8	8	7	8	8	8	8	8	8
133	7	8	8	8	8	7	7	7	7	7	7	7	7	8	8	7	7	7	7	8
134	8	8	8	8	8	8	9	11	13	14	12	9	8	8	7	8	8	8	8	8
135	7	7	7	7	7	11	16	19	21	21	20	17	12	9	8	7	7	8	8	7
136	8	8	8	8	12	18	22	25	26	27	26	23	19	12	10	9	8	8	8	8
137	8	7	7	9	15	21	25	28	30	31	29	26	23	17	11	9	8	8	8	8
138	8	8	8	9	16	23	27	32	38	38	33	30	25	19	13	9	8	8	8	8
139	8	8	7	9	16	22	26	29	31	31	30	27	23	19	12	9	9	8	8	8
140	9	8	8	9	13	19	24	26	28	28	26	25	21	16	11	10	9	9	9	9
141	8	8	8	8	10	14	18	22	23	24	23	20	16	11	9	8	8	8	8	8
142	8	9	9	9	10	11	13	15	16	16	16	13	11	10	9	9	9	9	9	9
143	8	8	8	8	8	8	9	10	10	11	10	9	9	8	8	8	8	8	8	8
144	9	9	9	9	9	9	9	9	10	10	9	10	9	9	9	9	8	9	9	9
145	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	9
146	9	9	8	9	9	9	9	8	8	8	8	8	8	9	8	8	8	8	8	9

Table 4.1 (d) Gray-Scale Map of RED Ball

y-pixel \ x-pixel	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184
130	8	8	8	8	8	9	9	9	9	9	9	10	10	9	10	9	9	9	9	9
131	8	8	7	7	8	8	8	9	9	9	9	9	8	9	8	8	9	9	8	8
132	8	8	8	8	8	8	8	8	8	9	9	8	9	9	8	8	8	8	8	8
133	8	8	8	8	7	8	8	7	8	8	8	8	8	8	8	8	8	8	8	8
134	8	8	8	7	7	7	7	7	7	7	7	7	7	8	8	8	8	8	8	9
135	7	7	7	7	7	6	8	9	9	10	9	8	8	7	7	8	7	7	8	8
136	8	8	8	8	7	9	11	13	14	14	14	12	10	8	8	8	8	8	8	8
137	7	8	7	6	8	10	13	16	18	18	16	14	11	9	7	7	8	8	7	8
138	8	8	7	7	9	12	16	22	29	30	23	18	16	10	8	8	8	8	8	8
139	8	7	7	6	7	12	14	18	21	22	19	17	14	10	8	7	7	8	9	9
140	8	9	8	8	9	11	13	16	17	17	17	15	12	10	9	9	9	9	9	9
141	9	8	8	8	7	8	11	12	13	14	13	11	10	8	7	8	8	8	8	8
142	9	9	9	9	9	9	10	10	11	11	11	10	9	9	9	9	9	9	9	9
143	8	8	8	8	8	8	8	9	8	8	9	9	8	9	9	9	9	9	9	9
144	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
145	9	9	9	9	8	8	8	8	8	9	9	10	9	9	9	9	9	9	9	9
146	8	9	9	9	9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9

Table 4.1 (e) Gray-Scale Map of BROWN Ball

y-pixel \ x-pixel	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204
130	9	9	9	10	10	9	10	10	9	10	10	10	10	10	9	10	10	9	10	10
131	8	8	9	8	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
132	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	10	9	9	10
133	8	8	8	9	9	9	9	9	8	8	8	8	9	9	9	9	9	9	9	9
134	10	8	8	10	8	8	8	7	7	7	7	8	8	8	9	9	9	9	9	9
135	8	8	8	7	7	6	6	6	6	6	7	6	6	7	7	8	8	8	8	9
136	9	9	9	9	7	7	8	9	9	10	9	9	8	7	8	9	9	9	9	9
137	8	8	8	7	6	7	8	10	12	14	12	10	8	7	7	8	8	9	9	9
138	8	8	7	6	6	7	9	15	25	31	24	17	14	8	8	8	8	9	9	10
139	9	9	7	7	6	7	9	13	18	21	18	14	11	8	7	8	8	8	9	9
140	9	9	9	8	7	8	9	10	13	14	13	11	9	8	8	8	9	10	10	10
141	8	8	8	7	7	7	7	9	10	10	10	8	7	7	8	8	9	9	9	9
142	10	10	9	10	8	8	8	8	8	8	8	8	9	9	9	10	10	10	10	10
143	9	9	9	9	8	8	8	8	8	8	8	8	8	9	9	9	10	10	10	10
144	9	9	9	9	9	9	9	10	9	9	10	10	9	10	10	10	10	10	10	10
145	9	9	9	9	9	9	9	9	9	9	9	10	10	10	10	10	10	10	10	10
146	9	9	9	9	9	10	10	9	9	9	9	10	10	10	10	10	10	10	10	10

Table 4.1 (f) Gray-Scale Map of BLUE Ball

x-pixel GREEN centre = (214,138)

y-pixel	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
130	9	10	9	9	10	10	10	10	10	10	10	11	10	11	11	10	10	10	10	11
131	9	9	9	9	9	9	9	9	9	10	10	9	10	10	9	9	9	9	9	9
132	9	9	9	9	9	9	9	9	9	10	10	9	10	10	9	9	9	9	9	9
133	9	9	9	9	9	9	8	8	8	8	9	9	9	10	9	9	9	9	9	9
134	9	9	9	10	9	8	8	7	6	6	6	6	8	9	9	9	9	9	9	9
135	8	9	9	8	7	7	5	5	5	5	5	5	5	6	7	8	8	9	9	9
136	10	10	9	9	8	6	6	6	6	6	7	6	6	7	7	8	9	9	9	10
137	9	9	8	8	6	3	4	6	6	10	10	7	7	6	5	7	8	8	8	8
138	9	9	9	7	6	6	6	8	17	27	28	20	13	9	6	8	8	9	9	8
139	9	9	9	7	7	5	5	7	12	18	19	14	9	8	6	7	8	8	8	8
140	10	10	9	9	7	6	6	6	8	11	11	9	8	7	7	8	9	9	9	9
141	9	9	9	9	7	7	5	6	7	8	8	7	6	6	7	8	8	9	9	7
142	10	11	11	10	10	9	7	7	7	7	7	7	7	8	9	9	10	10	9	10
143	10	10	10	10	10	9	9	8	8	8	8	8	8	8	9	9	9	9	9	9
144	10	10	10	10	10	10	10	10	10	10	9	10	10	9	10	10	10	10	10	10
145	10	11	10	10	10	10	9	10	10	10	10	10	10	10	10	10	10	9	9	10
146	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10

Table 4.1 (g) Gray-Scale Map of GREEN Ball

x-pixel BLACK centre = (235,138)

y-pixel	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245
130	10	10	10	9	10	10	9	9	9	9	9	9	9	9	10	10	10	9	9	9
131	9	9	9	9	9	9	9	9	9	9	9	9	9	9	8	9	9	8	9	9
132	10	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	8
133	9	9	9	9	10	9	8	8	8	8	8	8	8	8	8	8	9	9	9	9
134	9	9	9	9	8	7	7	5	5	5	5	6	7	7	8	9	9	8	9	8
135	9	9	9	8	6	5	3	2	2	1	1	2	5	5	8	8	8	8	8	8
136	9	9	9	7	6	3	1	1	1	1	1	1	2	5	7	7	9	9	9	9
137	9	8	7	6	3	2	1	1	3	5	3	1	1	2	5	6	7	7	8	8
138	9	8	8	5	2	2	1	6	22	27	18	8	6	2	7	7	7	9	8	9
139	8	8	7	5	2	1	0	3	9	11	7	2	1	2	5	7	7	8	8	8
140	10	9	8	6	4	1	1	1	3	4	2	1	2	3	6	7	8	9	9	9
141	10	8	9	7	4	3	1	1	2	2	1	1	2	4	7	8	8	9	9	9
142	9	10	9	8	8	6	4	3	3	3	4	4	5	7	8	9	10	10	10	10
143	10	9	9	9	8	7	6	6	5	6	6	6	7	8	8	9	9	9	9	9
144	10	10	10	10	9	9	9	8	8	8	8	9	9	9	10	10	10	10	10	10
145	10	10	9	9	9	9	9	9	8	8	9	8	8	9	9	9	9	9	9	9
146	10	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

Table 4.1 (h) Gray-Scale Map of BLACK Ball

Trial Number	WHITE	YELLOW	PINK	RED	BROWN	BLUE	GREEN	BLACK
1	41.1	37.5	32.8	23.3	13.0	8.6	7.0	2.4
2	41.2	38.3	33.0	23.8	13.2	8.9	7.1	2.2
3	41.0	37.9	32.5	23.6	13.4	8.8	7.2	2.3
4	40.7	37.0	32.0	22.9	13.0	8.7	6.8	2.8
5	40.8	37.0	31.6	23.3	12.9	8.5	7.4	2.9
6	41.0	36.5	31.5	22.7	13.0	9.2	6.9	2.3
7	40.8	36.9	31.9	23.4	13.0	8.8	7.3	3.5
8	41.6	37.7	31.9	23.2	12.7	8.6	7.0	3.3
9	40.9	37.7	31.9	23.4	13.3	9.0	7.0	2.4
10	40.7	37.4	32.1	23.0	13.2	8.7	7.0	2.2
11	40.7	37.7	32.9	23.2	13.1	8.5	7.0	2.1
12	41.1	38.3	33.5	23.8	13.1	9.0	7.3	2.1
13	40.8	38.1	32.9	23.6	13.5	9.2	7.6	2.3
14	40.9	37.5	32.3	23.2	12.9	8.7	7.0	2.4
15	41.0	37.0	32.2	23.1	12.6	8.8	7.0	2.1
16	40.8	36.6	32.3	22.8	13.2	8.6	7.0	3.0
17	41.1	37.3	32.0	23.2	13.0	8.6	6.5	2.3
18	41.1	37.1	32.3	23.3	13.0	8.6	7.0	3.2
19	40.4	37.5	32.8	23.3	13.5	9.0	7.0	3.4
20	40.8	37.7	32.7	23.2	13.3	9.0	7.4	3.0
21	41.0	37.2	32.7	22.7	12.7	8.5	6.3	2.5
22	41.4	37.6	32.8	23.7	13.3	8.9	7.0	2.8
23	40.9	37.7	33.1	23.2	13.2	9.3	7.0	2.8
24	40.9	36.3	32.1	22.9	12.5	8.6	7.0	2.5
25	40.7	36.8	31.7	23.0	12.8	8.6	7.0	2.7
26	40.5	36.9	32.4	23.2	12.9	8.2	7.0	2.8
27	41.1	37.0	32.0	23.2	12.7	8.4	7.0	3.3
28	41.0	36.9	32.1	22.9	12.8	8.4	7.0	3.1
29	41.1	37.2	32.3	23.6	13.2	9.3	7.2	3.5
30	40.9	37.3	32.2	23.3	13.3	8.8	7.5	3.5
31	41.1	37.7	32.8	23.2	13.1	8.5	6.9	1.9
32	41.1	37.3	33.3	23.9	13.3	8.7	7.1	2.9
33	41.3	37.2	33.0	23.6	13.2	9.0	7.3	2.0
34	41.1	37.2	31.7	23.1	12.3	8.9	6.7	2.4
35	40.9	37.1	31.7	22.8	12.8	8.3	6.8	2.6
36	41.0	36.7	31.4	22.9	13.0	8.7	6.9	2.7
37	40.7	36.9	32.2	23.0	12.9	8.3	6.7	2.7
38	41.1	36.6	32.0	23.2	13.2	8.5	6.6	2.8
39	41.3	36.9	32.7	23.4	13.4	8.7	6.7	2.4
40	41.4	37.5	32.3	23.6	13.3	8.8	6.7	3.3
41	41.1	38.2	32.8	23.6	12.9	8.9	6.7	1.8
42	40.9	38.4	32.8	23.9	13.3	8.9	6.9	2.5
43	41.2	37.8	32.2	23.0	13.1	8.7	7.0	2.2
44	40.6	37.7	32.2	23.3	12.8	8.8	6.7	2.9
45	40.9	37.2	31.7	22.8	13.0	8.9	6.9	2.5
46	41.1	37.4	32.0	23.1	13.3	8.6	7.0	2.5
47	40.7	37.1	31.1	23.3	12.9	8.6	6.9	2.6
48	40.7	37.1	31.7	22.9	13.3	8.2	7.0	2.1
49	41.2	37.3	31.8	23.3	13.3	8.6	7.0	2.6
50	41.3	37.8	33.0	23.5	13.5	8.5	6.9	2.0

Table 4.2 (a) Ball Colours and Their Corresponding Average Gray-Scale Values

Trial Number	WHITE	YELLOW	PINK	RED	BROWN	BLUE	GREEN	BLACK
51	40.8	38.0	32.6	23.7	12.7	8.9	6.8	2.0
52	41.2	37.9	33.3	23.7	13.7	9.0	7.0	2.1
53	41.4	38.0	32.0	22.9	13.2	8.7	7.2	2.2
54	41.2	37.5	31.8	23.3	13.1	8.4	7.0	2.2
55	40.6	37.0	31.5	22.6	13.1	8.5	7.1	2.7
56	40.8	37.4	31.2	23.0	13.0	9.0	7.0	2.8
57	40.5	37.3	31.4	22.9	12.9	8.4	6.7	2.4
58	40.1	37.1	31.9	23.0	13.1	8.5	6.8	2.5
59	41.1	37.7	33.0	23.4	13.5	8.7	6.9	2.0
60	41.2	37.6	32.7	23.2	13.5	8.7	7.0	2.3
61	41.2	37.9	32.7	23.4	13.1	8.8	7.0	2.0
62	41.4	38.6	33.2	23.5	14.0	8.8	7.0	1.9
63	41.3	38.3	32.8	23.3	13.1	8.8	7.0	2.0
64	41.1	37.3	32.1	22.8	13.3	8.5	7.4	2.1
65	40.6	37.1	32.2	22.6	13.3	9.0	7.0	2.4
66	41.0	37.5	32.0	22.7	13.1	9.0	7.2	3.4
67	40.9	37.9	31.9	22.6	13.1	8.5	6.9	2.7
68	40.9	37.1	31.8	22.6	13.2	8.4	6.7	2.0
69	41.0	37.2	32.4	22.7	13.4	8.9	7.2	2.5
70	41.4	36.9	32.3	23.4	13.2	8.8	6.9	2.5
71	41.3	38.7	33.5	23.3	13.4	9.0	6.8	1.7
72	41.2	38.9	34.0	23.9	13.9	9.2	7.2	2.0
73	41.2	38.1	33.2	23.5	13.6	8.6	7.2	2.2
74	41.1	37.8	33.1	23.4	13.3	9.0	7.5	2.4
75	41.0	36.6	32.7	23.0	12.9	8.8	7.0	2.8
76	41.2	37.0	32.0	23.0	13.2	9.0	7.1	2.6
77	40.8	37.2	32.1	23.1	13.0	8.8	7.0	2.0
78	41.0	37.0	32.3	22.8	12.9	8.8	7.3	2.1
79	40.8	37.5	33.0	23.6	13.8	9.1	7.1	2.9
80	41.5	38.0	33.0	23.5	13.6	9.4	6.9	2.7
81	41.2	38.4	32.8	23.8	13.6	8.6	6.7	2.2
82	41.2	38.8	33.7	24.1	13.7	9.4	7.1	2.0
83	41.6	37.9	33.3	23.8	13.3	9.0	7.0	2.4
84	41.6	37.8	32.6	23.7	13.4	9.5	7.2	2.4
85	41.2	37.8	32.4	23.3	13.5	9.0	7.0	2.3
86	41.4	37.5	31.8	23.1	13.4	9.0	7.1	2.5
87	40.8	37.4	32.0	23.0	12.6	8.9	7.0	2.2
88	40.6	37.8	32.1	23.0	12.8	8.5	6.5	2.1
89	41.2	38.2	32.9	23.4	13.4	9.4	7.3	2.4
90	41.6	38.0	32.9	23.3	13.5	9.2	7.0	2.6
91	41.5	38.6	32.6	23.5	13.4	8.4	7.2	2.0
92	41.2	38.3	33.6	23.8	13.2	9.2	6.6	2.0
93	40.9	38.0	32.5	23.3	13.5	9.0	6.9	2.1
94	41.1	38.3	32.5	23.4	13.4	8.8	7.2	2.0
95	40.9	38.2	32.4	23.0	12.9	8.9	7.1	2.0
96	41.3	37.6	32.4	23.4	13.3	8.6	6.9	2.0
97	40.7	37.1	31.7	22.8	12.9	8.7	6.4	2.0
98	40.8	37.2	31.3	22.6	12.7	8.7	6.4	2.1
99	40.7	37.6	32.3	23.1	13.2	8.8	7.0	2.1
100	40.9	37.5	32.3	23.5	13.2	8.9	6.8	2.2

Table 4.2 (b) Ball Colours and Their Corresponding Average Gray-Scale Values

	MIN	MAX	MEAN	SD
WHITE	40.10	41.60	41.02	0.28
YELLOW	36.30	38.90	37.53	0.55
PINK	31.10	34.00	32.39	0.59
RED	22.60	24.10	23.24	0.35
BROWN	12.30	14.00	13.16	0.30
BLUE	8.20	9.50	8.78	0.28
GREEN	6.30	7.60	6.98	0.23
BLACK	1.70	3.50	2.45	0.43

Table 4.3 Colour Gray-Scale Sample Mean and Standard Deviation

COLOUR	LOW THRESHOLD	HIGH THRESHOLD
WHITE	39.8	42.4
YELLOW	35.1	39.8
PINK	29.4	35.1
RED	21.5	25.0
BROWN	11.7	14.7
BLUE	7.8	10.2
GREEN	5.8	7.8
BLACK	0.3	4.6

Table 4.4 Low and High Colour Thresholds

9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
8	8	8	8	9	9	9	9	9	8	8	8	9	11	14	15	15	12	10	8	8	8	8	8	9
9	9	9	9	9	9	9	9	9	9	9	11	16	21	23	23	23	22	18	14	11	9	9	9	9
9	9	9	9	9	9	9	9	9	9	10	16	21	24	26	28	28	26	23	19	13	10	9	9	9
9	9	9	9	9	9	9	9	8	9	13	20	25	29	34	37	35	31	27	22	16	12	10	10	9
8	9	9	9	9	9	9	9	9	9	14	20	25	30	37	40	38	32	28	23	17	12	10	10	9
9	9	9	9	9	9	9	9	9	9	13	20	25	28	30	31	30	29	26	21	16	12	10	10	10
8	8	9	9	9	9	11	14	15	15	16	20	23	26	28	28	27	26	23	18	13	10	9	9	9
9	9	9	9	11	17	20	23	24	24	24	23	23	23	23	23	23	21	17	15	11	10	10	10	9
9	9	9	10	16	21	25	27	28	28	27	25	22	18	17	17	16	14	12	10	9	9	9	9	9
9	9	9	13	19	25	28	33	36	34	31	28	23	18	15	13	12	11	10	10	9	9	9	10	9
8	9	9	13	20	26	30	37	40	37	33	29	23	18	13	11	11	10	10	9	9	9	9	9	9
9	9	9	13	19	24	28	30	31	31	29	26	22	16	12	10	10	10	10	10	9	9	9	9	9
9	9	9	11	17	22	25	27	28	28	26	23	19	14	10	10	10	10	9	9	9	9	9	9	8
9	9	9	10	13	17	20	23	23	23	21	18	14	11	10	10	10	10	10	10	10	10	10	9	9
9	9	9	9	10	11	13	15	16	16	14	12	11	10	9	9	9	9	9	9	9	9	9	9	9
9	9	10	10	10	10	11	12	12	12	11	11	10	10	10	9	9	9	9	9	10	10	10	10	10
9	9	9	9	9	9	10	10	10	10	10	10	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	10	10	10	10	10	10	10	10	10	9	9	9	9	9	10	9	9	9	10	10
9	9	9	9	9	9	9	9	9	9	7	8	9	8	8	8	8	9	9	9	9	9	9	9	9

Table 4.5 (a) Pixel Map of 2 Touching Red Balls, Threshold = 19

9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
8	8	8	8	9	9	9	9	9	8	8	8	9	11	14	15	15	12	10	8	8	8	8	9
9	9	9	9	9	9	9	9	9	9	9	11	16	21	23	23	23	22	18	14	11	9	9	9
9	9	9	9	9	9	9	9	9	9	10	16	21	24	26	28	28	26	23	19	13	10	9	9
9	9	9	9	9	9	9	9	8	9	13	20	25	29	34	37	35	31	27	22	16	12	10	10
8	9	9	9	9	9	9	9	9	9	14	20	25	30	37	40	38	32	28	23	17	12	10	10
9	9	9	9	9	9	9	9	9	9	13	20	25	28	30	31	30	29	26	21	16	12	10	10
8	8	9	9	9	9	11	14	15	15	16	20	23	26	28	28	27	26	23	18	13	10	9	9
9	9	9	9	11	17	20	23	24	24	24	23	23	23	23	23	23	21	17	15	11	10	10	9
9	9	9	10	16	21	25	27	28	28	27	25	22	18	17	17	16	14	12	10	9	9	9	9
9	9	9	13	19	25	28	33	36	34	31	28	23	18	15	13	12	11	10	10	9	9	9	9
8	9	9	13	20	26	30	37	40	37	33	29	23	18	13	11	11	10	10	9	9	9	9	9
9	9	9	13	19	24	28	30	31	31	29	26	22	16	12	10	10	10	10	10	9	9	9	9
9	9	9	11	17	22	25	27	28	28	26	23	19	14	10	10	10	10	9	9	9	9	9	8
9	9	9	10	13	17	20	23	23	23	21	18	14	11	10	10	10	10	10	10	10	10	10	9	9
9	9	9	9	10	11	13	15	16	16	14	12	11	10	9	9	9	9	9	9	9	9	9	9
9	9	10	10	10	10	11	12	12	12	11	11	10	10	10	9	9	9	9	9	10	10	10	10	10
9	9	9	9	9	9	10	10	10	10	10	10	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	10	10	10	10	10	10	10	10	10	9	9	9	9	9	10	9	9	9	10	10
9	9	9	9	9	9	9	9	9	9	7	8	9	8	8	8	8	9	9	9	9	9	9	9	9

Table 4.5 (b) Pixel Map of 2 Touching Red Balls, Threshold = 20

9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
8	8	8	8	9	9	9	9	9	8	8	8	9	11	14	15	15	12	10	8	8	8	8	8	9
9	9	9	9	9	9	9	9	9	9	9	11	16	21	23	23	23	22	18	14	11	9	9	9	9
9	9	9	9	9	9	9	9	9	9	10	16	21	24	26	28	28	26	23	19	13	10	9	9	9
9	9	9	9	9	9	9	9	8	9	13	20	25	29	34	37	35	31	27	22	16	12	10	10	9
8	9	9	9	9	9	9	9	9	9	14	20	25	30	37	40	38	32	28	23	17	12	10	10	9
9	9	9	9	9	9	9	9	9	9	13	20	25	28	30	31	30	29	26	21	16	12	10	10	10
8	8	9	9	9	9	11	14	15	15	16	20	23	26	28	28	27	26	23	18	13	10	9	9	9
9	9	9	9	11	17	20	23	24	24	24	23	23	23	23	23	23	21	17	15	11	10	10	10	9
9	9	9	10	16	21	25	27	28	28	27	25	22	18	17	17	16	14	12	10	9	9	9	9	9
9	9	9	13	19	25	28	33	36	34	31	28	23	18	15	13	12	11	10	10	9	9	9	10	9
8	9	9	13	20	26	30	37	40	37	33	29	23	18	13	11	11	10	10	9	9	9	9	9	9
9	9	9	13	19	24	28	30	31	31	29	26	22	16	12	10	10	10	10	10	9	9	9	9	9
9	9	9	11	17	22	25	27	28	28	26	23	19	14	10	10	10	10	9	9	9	9	9	9	8
9	9	9	10	13	17	20	23	23	23	21	18	14	11	10	10	10	10	10	10	10	10	10	9	9
9	9	9	9	10	11	13	15	16	16	14	12	11	10	9	9	9	9	9	9	9	9	9	9	9
9	9	10	10	10	10	11	12	12	12	11	11	10	10	10	9	9	9	9	9	10	10	10	10	10
9	9	9	9	9	9	10	10	10	10	10	10	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	10	10	10	10	10	10	10	10	10	9	9	9	9	9	10	9	9	9	10	10
9	9	9	9	9	9	9	9	9	9	7	8	9	8	8	8	8	9	9	9	9	9	9	9	9

Table 4.5 (c) Pixel Map of 2 Touching Red Balls, Threshold = 21

9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
8	8	8	8	9	9	9	9	9	8	8	8	9	11	14	15	15	12	10	8	8	8	8	9
9	9	9	9	9	9	9	9	9	9	9	11	16	21	23	23	23	22	18	14	11	9	9	9
9	9	9	9	9	9	9	9	9	9	10	16	21	24	26	28	28	26	23	19	13	10	9	9
9	9	9	9	9	9	9	9	8	9	13	20	25	29	34	37	35	31	27	22	16	12	10	9
8	9	9	9	9	9	9	9	9	9	14	20	25	30	37	40	38	32	28	23	17	12	10	9
9	9	9	9	9	9	9	9	9	9	13	20	25	28	30	31	30	29	26	21	16	12	10	10
8	8	9	9	9	9	11	14	15	15	16	20	23	26	28	28	27	26	23	18	13	10	9	9
9	9	9	9	11	17	20	23	24	24	24	23	23	23	23	23	23	21	17	15	11	10	10	9
9	9	9	10	16	21	25	27	28	28	27	25	22	18	17	17	16	14	12	10	9	9	9	9
9	9	9	13	19	25	28	33	36	34	31	28	23	18	15	13	12	11	10	10	9	9	9	9
8	9	9	13	20	26	30	37	40	37	33	29	23	18	13	11	11	10	10	9	9	9	9	9
9	9	9	13	19	24	28	30	31	31	29	26	22	16	12	10	10	10	10	10	9	9	9	9
9	9	9	11	17	22	25	27	28	28	26	23	19	14	10	10	10	10	9	9	9	9	9	8
9	9	9	10	13	17	20	23	23	23	21	18	14	11	10	10	10	10	10	10	10	10	9	9
9	9	9	9	10	11	13	15	16	16	14	12	11	10	9	9	9	9	9	9	9	9	9	9
9	9	10	10	10	10	11	12	12	12	11	11	10	10	10	9	9	9	9	9	10	10	10	10	10
9	9	9	9	9	9	10	10	10	10	10	10	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	10	10	10	10	10	10	10	10	10	9	9	9	9	9	10	9	9	9	10	10
9	9	9	9	9	9	9	9	9	9	7	8	9	8	8	8	8	9	9	9	9	9	9	9	9

Table 4.5 (d) Pixel Map of 2 Touching Red Balls, Threshold = 22

9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
8	8	8	8	9	9	9	9	9	8	8	8	9	11	14	15	15	12	10	8	8	8	8	8	9
9	9	9	9	9	9	9	9	9	9	9	11	16	21	23	23	23	22	18	14	11	9	9	9	9
9	9	9	9	9	9	9	9	9	9	10	16	21	24	26	28	28	26	23	19	13	10	9	9	9
9	9	9	9	9	9	9	9	8	9	13	20	25	29	34	37	35	31	27	22	16	12	10	10	9
8	9	9	9	9	9	9	9	9	9	14	20	25	30	37	40	38	32	28	23	17	12	10	10	9
9	9	9	9	9	9	9	9	9	9	13	20	25	28	30	31	30	29	26	21	16	12	10	10	10
8	8	9	9	9	9	11	14	15	15	16	20	23	26	28	28	27	26	23	18	13	10	9	9	9
9	9	9	9	11	17	20	23	24	24	24	23	23	23	23	23	23	21	17	15	11	10	10	10	9
9	9	9	10	16	21	25	27	28	28	27	25	22	18	17	17	16	14	12	10	9	9	9	9	9
9	9	9	13	19	25	28	33	36	34	31	28	23	18	15	13	12	11	10	10	9	9	9	10	9
8	9	9	13	20	26	30	37	40	37	33	29	23	18	13	11	11	10	10	9	9	9	9	9	9
9	9	9	13	19	24	28	30	31	31	29	26	22	16	12	10	10	10	10	10	9	9	9	9	9
9	9	9	11	17	22	25	27	28	28	26	23	19	14	10	10	10	10	9	9	9	9	9	9	8
9	9	9	10	13	17	20	23	23	23	21	18	14	11	10	10	10	10	10	10	10	10	10	9	9
9	9	9	9	10	11	13	15	16	16	14	12	11	10	9	9	9	9	9	9	9	9	9	9	9
9	9	10	10	10	10	11	12	12	12	11	11	10	10	10	9	9	9	9	9	10	10	10	10	10
9	9	9	9	9	9	10	10	10	10	10	10	9	9	9	9	9	9	9	9	9	9	9	9	9
9	9	9	9	9	10	10	10	10	10	10	10	10	10	9	9	9	9	9	10	9	9	9	10	10
9	9	9	9	9	9	9	9	9	9	7	8	9	8	8	8	8	9	9	9	9	9	9	9	9

Table 4.5 (e) Pixel Map of 2 Touching Red Balls, Threshold = 23

	BLACK	WHITE
RED	0	1000
YELLOW	0	1100
GREEN	0	900
BROWN	0	1000
BLUE	0	950
PINK	0	1100
BLACK	0	700
WHITE	0	1000

Table 4.6 On-board Camera Image Digitisation Settings

	High Threshold (See note 1)	Low Threshold (See note 2)	Mode (POSITIVE/NEGATIVE)
RED	25	19	POSITIVE
YELLOW	28	24	POSITIVE
GREEN	34	20	POSITIVE
BROWN	28	22	POSITIVE
BLUE	32	18	POSITIVE
PINK	26	22	POSITIVE
BLACK	4	0	NEGATIVE
WHITE	26	22	POSITIVE

Note :
1. High threshold is loaded initially for segmentation.
2. Low threshold is loaded once deviation between expected and actual ball centres drop below a limit.

Table 4.7 On-board Camera Threshold and Segmentation Mode Settings

Trial Number	(*x1*, *y1*)	(*x2*, *y2*)	(X_C, Y_C)
1	(-538.31, 32.94)	(-544.66, -73.69)	(-3.175, -53.315)
2	(-493.66, -140.59)	(-500.22, -247.28)	(-3.280, -53.345)
3	(-414.56, 280.63)	(-421.19, 174.91)	(-3.315, -52.860)
4	(-393.59, 178.56)	(-399.53, 72.63)	(-2.970, -52.965)
5	(-320.22, -169.03)	(-327.50, -276.25)	(-3.640, -53.610)
6	(-473.59, 56.53)	(-480.63, -49.91)	(-3.520, -53.220)
7	(-529.38, 142.16)	(-535.78, 35.78)	(-3.200, -52.690)
8	(-465.53, -41.03)	(-471.53, -147.81)	(-3.000, -53.390)
9	(-535.75, 64.47)	(-541.84, -41.97)	(-3.045, -53.220)
10	(-475.84, 137.00)	(-482.31, 30.78)	(-3.235, -53.110)

MEAN (X_C, Y_C) = (-3.238, -53.173)

Note : (*Xc*, *Yc*) = ((*x2*-*x1*)/2, (*y2*-*y1*)/2))

Table 4.8 On-board Camera Tool Calibration Results

Chapter 5 Shot Identification and Shot Difficulty Index

In previous chapters, the mathematics of snooker and the computation of ball positions by machine vision have been studied. An algorithm based on a series of geometric tests to determine whether a ball can be potted is presented herein. In any particular game situation, there could be more than 1 shot that can be played. Hence a method of judging the relative difficulties of various shots so that they can be directly compared has to be devised. This ability to sort a selection of shots by their relative difficulties is most important, without which there will be no guidance to the choice of shots. The combined difficulties of the current shot and the anticipated subsequent shot is the deciding factor in the choice of the best 1-step ahead game plan.

5.1 Identification of Feasible Shots

Given the (x, y) world co-ordinates of a pocket, a target ball, and a striking ball, the objective is to determine whether the striking ball can pot the target ball into a pocket. In chapter 3, where the path of the striking ball after impact was studied, the path of the target ball was largely ignored. This is because the imaginary ball technique is used to determine shot angles. The technique is very popular among snooker players and is based on the assumption that snooker balls are perfectly smooth spheres with negligible friction during collision between a stationary target ball and a striking ball. As a result, the target ball will travel in the direction of the line joining the target ball centre and the striking ball centre at the moment of impact.

To apply the imaginary ball technique to the shot feasibility test, the world co-ordinates of the imaginary ball has to be derived from the pocket and target ball positions. By definition the distance between the target ball and the imaginary ball must be D, the snooker ball diameter, since they are touching each other at the moment of impact. As the pocket, target ball and imaginary ball are co-linear, the world co-ordinates of the imaginary ball can be computed by a translation followed by a rotation.

Let the angle between the direction vector from the target ball to the pocket and the positive world Y-axis be θ (Note that the sense of the rotation is crucial and that throughout the system all clockwise rotation are treated as positive, and all angles are measured relative to the direction vector parallel to the positive world Y-axis unless otherwise stated, see figure 5.1. Suppose the world co-ordinates of the target ball are (T_x, T_y), then the co-ordinates of the imaginary ball is first set as $(T_x, T_y - D)$, which is equivalent to translating the target ball by $-D$ in the Y-direction. To find the true co-ordinates of the imaginary ball, the point $(T_x, T_y - D)$ is rotated by θ about the target

ball, the resulting point being (IMG_x, IMG_y), which is where the striking ball should be at the moment of impact. This sequence of operations is summed up in figure 5.2.

Having calculated the point (IMG_x, IMG_y), it becomes relatively straightforward to test whether the target ball can be potted by a striking ball at (S_x, S_y) in world co-ordinates. This involves translating the human skill of sighting a shot into a rigid set of geometric tests. Basically, a human player first assesses whether the striking ball is able to hit the target ball so as to send it towards the pocket. Suppose the line *L1* passes through the pocket and target ball and line *L2* is perpendicular to *L1* through the imaginary ball, as in figure 5.3. It follows that a necessary condition for the target ball to be potted is that the pocket and the striking ball have to be on opposite sides of the line *L2*. If this condition is met, the player will then align his line of sight with the line joining the pocket and the target ball to see if any ball is obstructing the path. This visual detection of obstruction is then repeated upon the line joining the striking ball and the imaginary ball. The shot is feasible if and only if both paths are clear of any obstacles.

The shot feasibility test adopted in the system mirrors the above human skill and is broken down into 3 simple tests to detect obstructions to the striking ball path, impact between striking ball and target ball, and the target ball path respectively.

<u>Shot Feasibility Test 1</u> : To determine if the striking ball path is clear.

Given the striking ball co-ordinates (S_x, S_y) and imaginary ball co-ordinates (IMG_x, IMG_y) as input to this test, the objective is to find the number of balls (remembering there can be a maximum of 10 red balls and 7 colour balls on the table) lying within the path defined by the points (S_x, S_y) and (IMG_x, IMG_y).

A preliminary test is first used to eliminate any ball that is nowhere near the path from consideration. This involves drawing a rectangle *A* as shown in figure 5.4. For every ball that is present on the table, if it lies outside rectangle *A* then it cannot possibly obstruct the path. This is a simple and effective test that should eliminate the majority of balls that cannot obstruct the path. Otherwise, the co-ordinates of the ball inside rectangle *A* are recorded. The output of the preliminary test is a set of co-ordinates representing the balls that may obstruct the path. This set of co-ordinates then becomes the input to the subsequent test.

Let *L1* be the line passing through the points (S_x, S_y) and (IMG_x, IMG_y) and that (B_x, B_y) is the co-ordinates of a ball inside rectangle *A*. A line *L2* passing through (B_x, B_y) perpendicular to *L1* can be drawn and *L1* and *L2* intersect at point *P1*. By expressing *P1* as

$$\begin{pmatrix} S_x \\ S_y \end{pmatrix} + k\begin{pmatrix} D_x \\ D_y \end{pmatrix} \quad \text{where} \quad \begin{pmatrix} D_x \\ D_y \end{pmatrix} = \begin{pmatrix} IMG_x - S_x \\ IMG_y - S_y \end{pmatrix} \qquad \textbf{Eq. (5.1)}$$

and knowing that the dot product of two normal vectors equals zero, the situation shown in figure 5.5 becomes true and therefore :

$$\left(\begin{pmatrix} B_x \\ B_y \end{pmatrix} - \begin{pmatrix} S_x + kD_x \\ S_y + kD_y \end{pmatrix}\right)\begin{pmatrix} D_x \\ D_y \end{pmatrix} = 0$$

$$\Rightarrow (B_x - S_x)D_x - kD_x^2 + (B_y - S_y)D_y - kD_y^2 = 0$$

$$\Rightarrow k = \frac{(B_x - S_x)D_x + (B_y - S_y)D_y}{D_x^2 + D_y^2}$$

The coefficient k indicates the whereabouts of the point $P1$ on $L1$. By definition, $P1$ lies between (S_x, S_y) and (IMG_x, IMG_y) if and only if $0 \le k \le 1$. If this condition is satisfied, k is substituted back into Eq. (*5.1*) to work out the co-ordinates of point $P1$. The distance between $P1$ and (B_x, B_y) is then calculated as $D1$. Finally, it can be formulated that the ball centred at (B_x, B_y) obstructs the path defined by (S_x, S_y) and (IMG_x, IMG_y) if:

Condition (*1*) : $0 \le k \le 1$ and

Condition (*2*) : $D1 \le D + 5$ where D = ball diameter in mm

are both TRUE.

An extra 5 mm is added to D as a safety margin because at this stage ball positions computed from the overhead camera image which are known to contain errors of up to 10 mm are used.

Because the path is defined by the striking ball which exists on the table and the imaginary ball that does not, the striking ball itself is bound to satisfy Conditions (*1*) and (*2*) above and so the number of obstructing balls is at least 1. If more than 1 obstructing ball are detected, the path must be obstructed and so the shot becomes unfeasible. It can

thus be concluded that this test succeeds if only 1 ball (the striking ball) lies inside the path.

Shot Feasibility Test 2 : To determine if the striking ball makes contact with the target ball alone.

Given the co-ordinates of the imaginary ball, the objective of this test is to determine the number of balls (excluding the striking ball) that are less than (D+5) mm from the imaginary ball, D being the ball diameter.

At the moment of impact, the co-ordinates of the striking ball will be (IMG_x, IMG_y). At this point, the striking ball will have made contact with any ball whose distance from (IMG_x, IMG_y) is less than (D + 5) mm. Again, to allow for vision error, a margin of 5mm is added to D. As a result, *for each ball present on the snooker table excluding the* striking ball, if the ball centre lies inside the circle C of radius (D+5) mm centred at (IMG_x, IMG_y), the number of balls close to the imaginary ball is incremented by 1. The striking ball is excluded from this test because the striking ball can be extremely close to the imaginary ball position in a feasible shot.

According to the definition of (IMG_x, IMG_y), the target ball is bound to be less than (D+5) mm from the imaginary ball and so will be counted. But if the count of balls inside circle C is greater than 1, the striking ball will not be able to hit the target ball directly and so the shot is not feasible. This test succeeds only if the target ball alone is inside circle C, as in figure 5.6.

Shot Feasibility Test 3 : To determine if the target ball path is clear.

This test is identical to Shot Feasibility Test 1 except for different input parameters specifying the path to be checked. The target ball has to be able to travel towards the pocket without encountering any obstructions. Let the co-ordinates of the pocket and target ball be (P_x, P_y) and (T_x, T_y) respectively.

Before (P_x, P_y) and (T_x, T_y) can be directly substituted into Shot Feasibility Test 1, (P_x, P_y) has to be transformed. Figure 5.7 shows a typical pocket position on the snooker table which is defined such that any ball anywhere on the table can go into the pocket. Figure 5.8 shows that a ball situated extremely close to the physical pocket (as oppose to the pocket point used in shot calculations) will escape detection if the line segment defined by (P_x, P_y) and (T_x, T_y) is checked for obstruction. To solve this problem, the point (P_x, P_y) is extended towards the pocket by the following method.

Let θ be the angle between the line segment defined by (P_x, P_y) and (T_x, T_y) and the positive world Y-axis. Then the pocket point is first translated along the Y-axis by D, the ball diameter, by modifying its co-ordinates as $(P_x, P_y + D)$. This modified pocket point is then rotated about (P_x, P_y) by angle θ, resulting in the point (P'_x, P'_y). This series of transformation is analogous to the computation of the imaginary ball position, see figure 5.2.

Now the input parameters of Shot Feasibility Test 1 can be substituted by (P'_x, P'_y) and (T_x, T_y). The test will reveal the number of balls obstructing the path defined by (P'_x, P'_y) and (T_x, T_y). Again, by definition, the target ball lies in the path and so the test succeeds if only one obstructing ball (i.e. the target ball itself) is detected.

For a shot to be considered feasible, it has to pass Test 1, 2, and 3 described above successfully. Figure 5.9 shows a sample collection of shots including explanations as to why they are feasible or otherwise. It has to be stressed that the tests described in this section only indicate the geometric feasibility of a shot. It will be shown the following chapter that some geometrically feasible shots cannot be taken because a cue force beyond the maximum delivered by the pneumatic cue is demanded.

5.2 Shot Difficulty Factors

In an earlier robot snooker player system, the angular tolerance of the cueing angle of a shot was computed as a guide to its chance of success. Figure 5.10 shows an example of the computation of cueing angle tolerance. Naturally, the chance of success of a shot is directly proportional to the angular tolerance. Indeed this approach can be adopted again in this system. However, an original approach, more similar to the way a human player assesses shot difficulty, involving judging angles and distances, is devised and implemented.

In a game of snooker, a human player subjectively judges the relative difficulty of various shots using general rule-of-thumbs. For example, in general the nearer a target ball is to a pocket, the easier it can be potted. The aim here is to incorporate these rule-of-thumbs into the system by measuring distances and angles, thereby arriving at an arbitrary index which allows the relative difficulties of different shots to be compared.

The following 4 factors are identified as having a direct bearing on the difficulty of a shot.

(1) Distance Travelled by Striking Ball Before Impact

This measures the distance the striking ball must travel before colliding with the target ball. All else being equal, the shot difficulty is directly proportional to this distance. The reasoning behind this proposition is that given an angular error of $\delta\theta$ from the intended striking ball path (which may arise from vision error or robot error), the actual deviation of the striking ball from the intended path increases as the ball travels further, as shown in figure 5.11. Since the striking ball path is represented by the line segment defined by the initial striking ball position and the imaginary ball position (computation of which was already described above), the distance between them can be calculated. This number is termed *SDF1* (short for *Shot Difficulty Factor 1*) which is normalised to range from 0 to 1 by dividing it by the greatest possible distance between 2 balls i.e. the length of the table diagonal.

(2) Distance Between Target Ball and Pocket

Similar to *SDF1*, the further the target ball has to travel before reaching the pocket, the larger the deviation from the intended target ball path will be as a result of an angular error of $\delta\theta$. For any particular shot, the co-ordinates of the pocket position and the target ball are known, allowing this factor, *SDF2*, to be calculated as the distance between them. Again, *SDF2* is normalised in the same way as *SDF1*.

(3) Angle Between Target Ball Entry to Pocket and Ideal Entry Line

For each pocket on the table there exists a unique line of ideal entry which bisects the pocket into two equal halves. Let θ be the angle between this line and the line defined by the pocket point and the target ball. Figure 5.12 shows how θ can be computed given the pocket and target ball positions. Given that the target ball is at a fixed distance from the pocket, the difficulty of potting the target ball into the pocket increases with θ. Appendix B contains a mathematical proof of this proposition. As the target ball path deviates from the ideal line of entry, θ increases, and so does the difficulty of potting it. The value of θ ranges from 0° to 45° for a corner pocket and 0° to 90° for a side pocket. The shot difficulty factor relating to the target ball entry to the pocket, *SDF3*, is therefore computed by dividing θ by 45° or 90°, depending on the type of pocket involved.

(4) Angle Between Striking Ball Path and Target Ball Path

Given the pocket, target ball, imaginary ball, and striking ball positions, the computation of θ, the angle between the striking ball path and target ball path, is shown in figure 5.13. For any feasible shot, θ must lie between 0° and 90°. Like *SDF3* above, the shot difficulty increases with θ. Here, some human expertise is introduced. It is known that most professional players prefer a half-ball shot, where $\theta = 30°$ approximately, instead of a full-ball shot where the pocket, target ball, and striking ball form a straight line. The reasoning behind this preference is that in a full-ball shot, the position of the striking ball after the shot is restricted to the line of impact unless drastic spinning of the striking ball was introduced. In a half-ball shot, however, the resting position of the striking ball can be more easily controlled by varying the cue force, which enhances the chance of finding a subsequent shot. Thus θ', the deviation from the preferred cut, can be computed as $|\theta - 30|°$ which ranges from 0° to 60° to incorporate this preference into the system. As a result, ***SDF4*** is computed by dividing θ' by 60°.

5.3 Shot Difficulty Index

Figure 5.14 is a summary of the computation of the 4 shot difficulty factors. For a feasible shot, having determined the 4 individual shot difficulty factors *SDF1*, *SDF2*, *SDF3*, and *SDF4*, they are to be combined to form a general index, *Shot Difficulty Index*, which reflects the relative difficulty of a shot. For each of the 4 *SDF's*, its value is normalised to range between 0 and 1, and that a value of 1 always indicates maximum difficulty. Hence ***SDI*** can be computed as :

$$SDI = k_1SDF1 + k_2SDF2 + k_3SDF3 + k_4SDF4 \qquad \textit{Eq. (5.2)}$$

$$\text{where } k_i > 0,\ i = \{1,2,3,4\}$$

Thus defined, the relative difficulty of a shot is directly proportional to the value of *SDI*. Each *SDF* in the Eq. *(5.2)* is multiplied by a weighting factor k_i before summation.

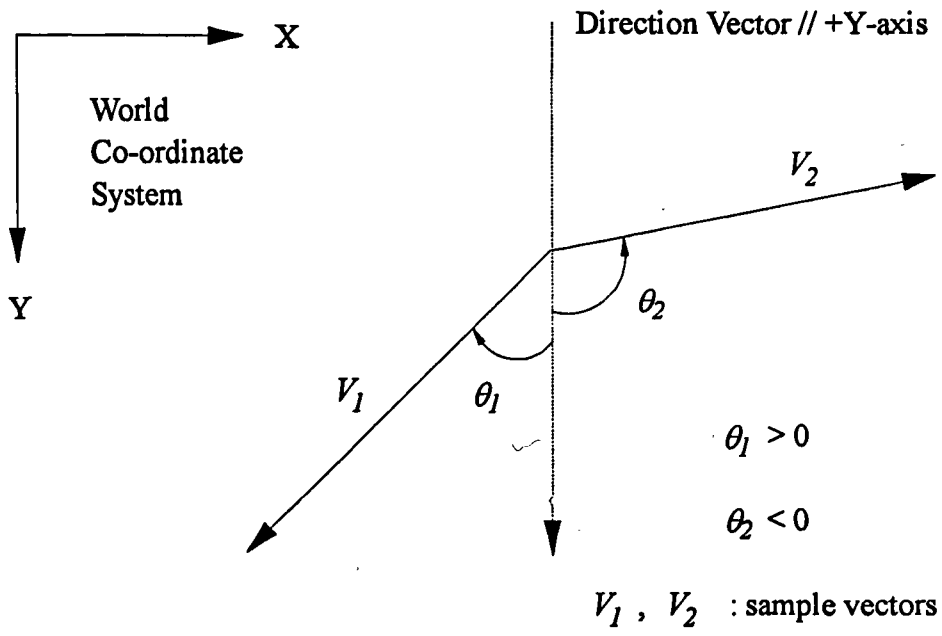
The effect of the values of k_i's on the value of *SDI* can be most clearly illustrated by an extreme example. Suppose k_1, k_3, and k_4 all equal 1 and k_2 equals 10000. The value of *SDI* will be dominated by (k_2SDF2), the second term in Eq. *(5.2)*, and the values of *SDF1*, *SDF3*, and *SDF4* becomes relatively negligible. Mathematically speaking, with the values of k_i's set as such, *SDI* tends to (k_2SDF2) as *SDF2* tends to 1. Practically speaking, this is equivalent to judging shot difficulty purely upon *SDF2*, the factor derived from the distance between the target ball and the pocket, while neglecting the other *SDF*s.

By careful selection of the values of k_i's, a level of system customisation can be achieved. In practice, the author found that by setting k_1, k_2 (the weightings on the factors on distances) as 3, and k_3, k_4 (the weightings on the factors on angles) as 1, the system's and the author's choice of shot among a selection tend to agree. This is in fact the key difference between the previous approach based on the calculation of cueing angle tolerance and this new approach. Had the previous approach been adopted, both customisation of this kind and the setting of a preferred angle of cut (see the definition of *SDF4* above) will not be possible.

5.4 Conclusions

An algorithm to determine whether a target ball can be potted by a striking ball (which has to be the cue ball in a game) has been designed and implemented. In case the target ball can be potted, an arbitrary index to the difficulty of potting this ball, *SDI*, can be calculated using an algorithm that works in a similar manner to the way a human player assesses shot difficulty.

Given any particular game situation, and suppose a red ball is to be potted, a simple double loop program structure can be used to check each red ball against each of the 6 pockets to check for shot feasibility, knowing the position of the cue ball. Every time a shot is found to be feasible, its associated difficulty *SDI* will be computed. This will result in a collection of feasible shots which can be sorted by their *SDI*'s. This forms the basis of a robot snooker player system that does not involve any kind of game-planning, like the previous system and an interim version of the current system. Figure 5.15 shows a digitised image of the snooker table with 6 red balls and the cue ball on it, and the best 5 feasible shots were highlighted on the display. The full list of feasible shots is tabulated in table 5.1 (a) and (b).



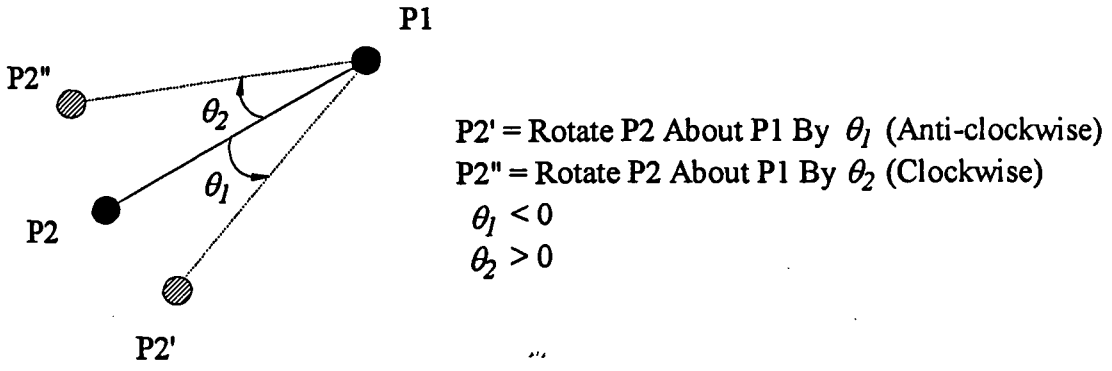
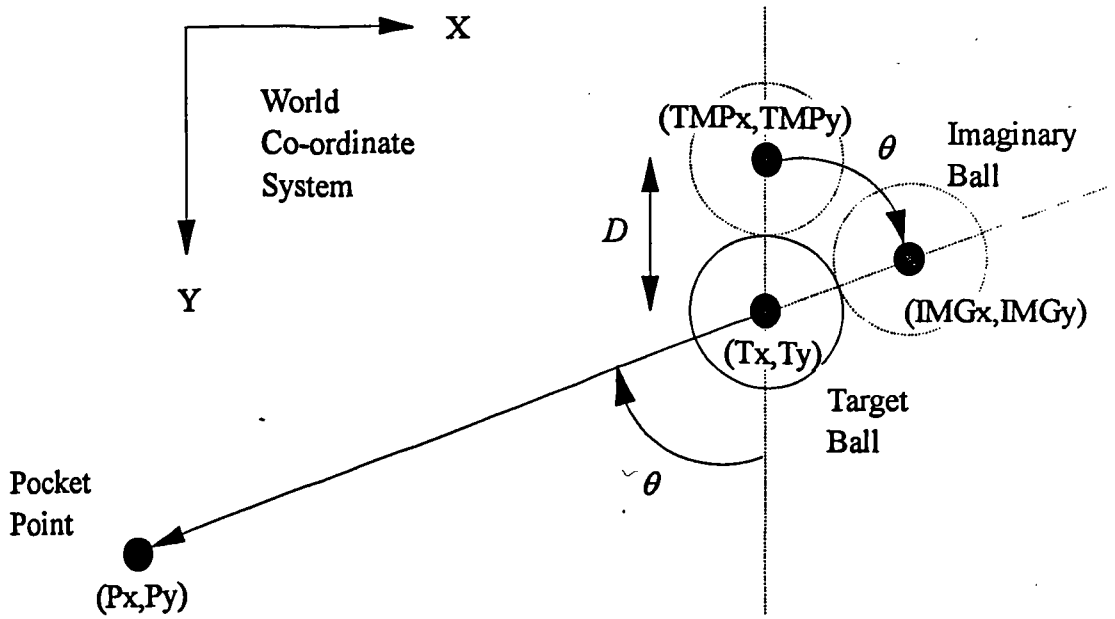


Figure 5.1 Senses of Angle Measurement and Rotation



Given :
(Tx, Ty) = Target Ball Position
(Px, Py) = Pocket Position
D = Ball Diameter

Compute :
θ = -atan (Px-Tx, Py-Ty)
(TMPx, TMPy) = (Tx, Ty-D)
(IMGx, IMGy) = Rotate (TMPx, TMPy) About (Tx, Ty) By θ

Figure 5.2 Computation of Imaginary Ball Position

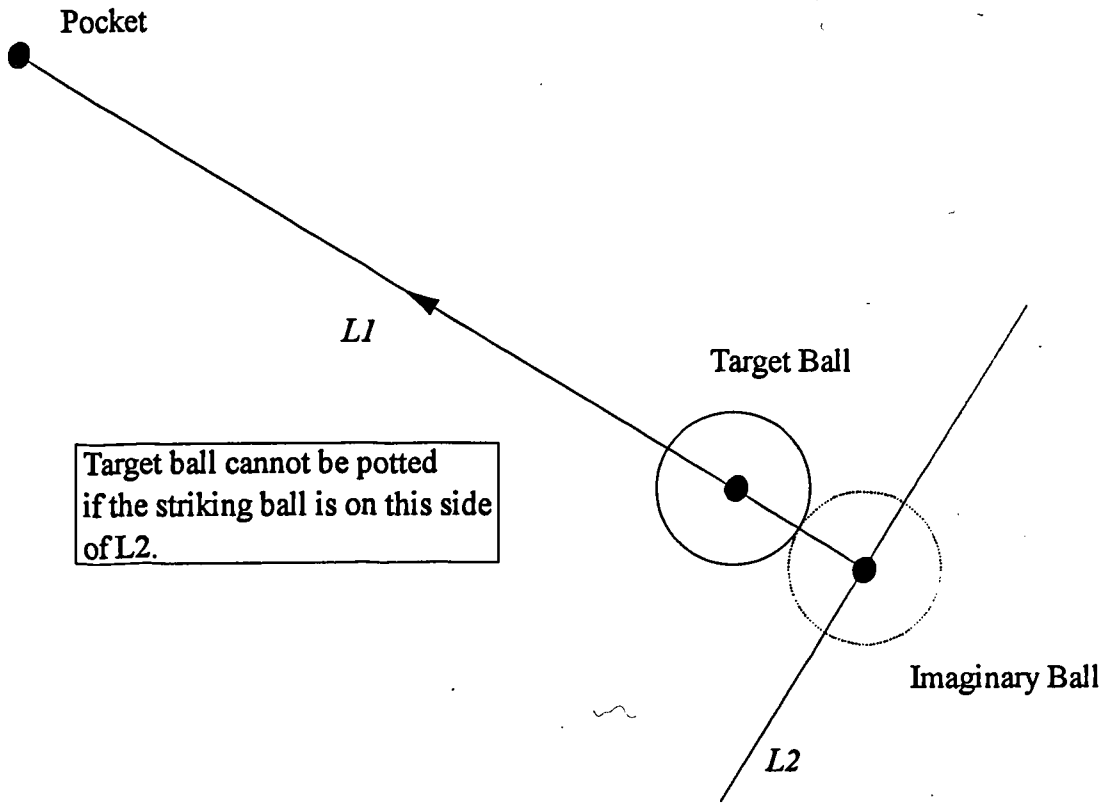
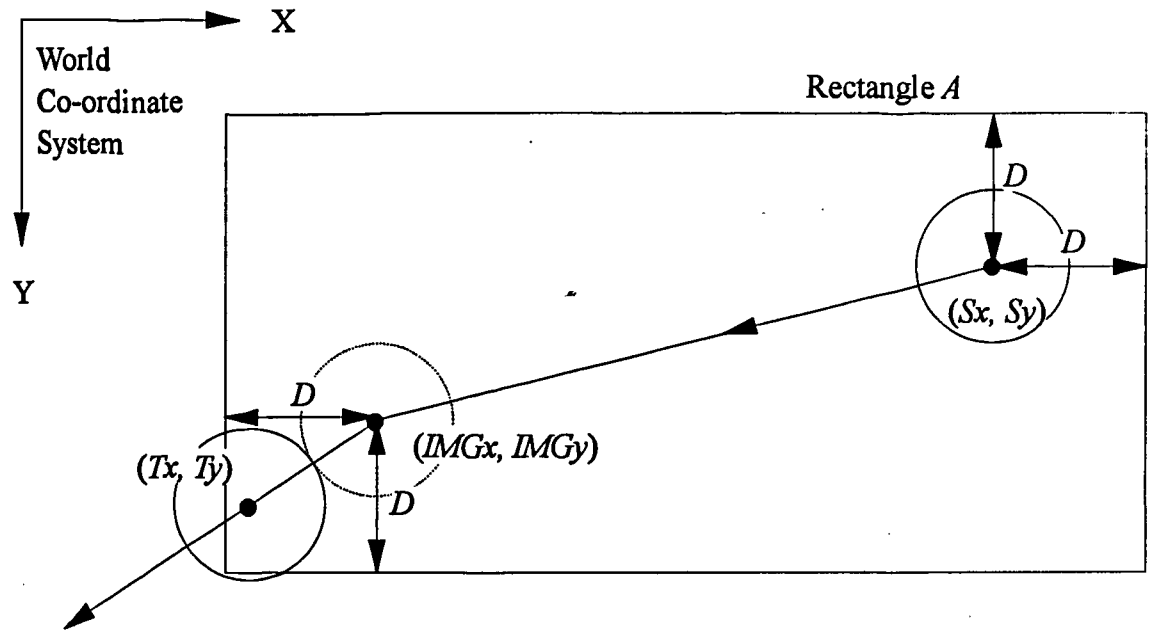


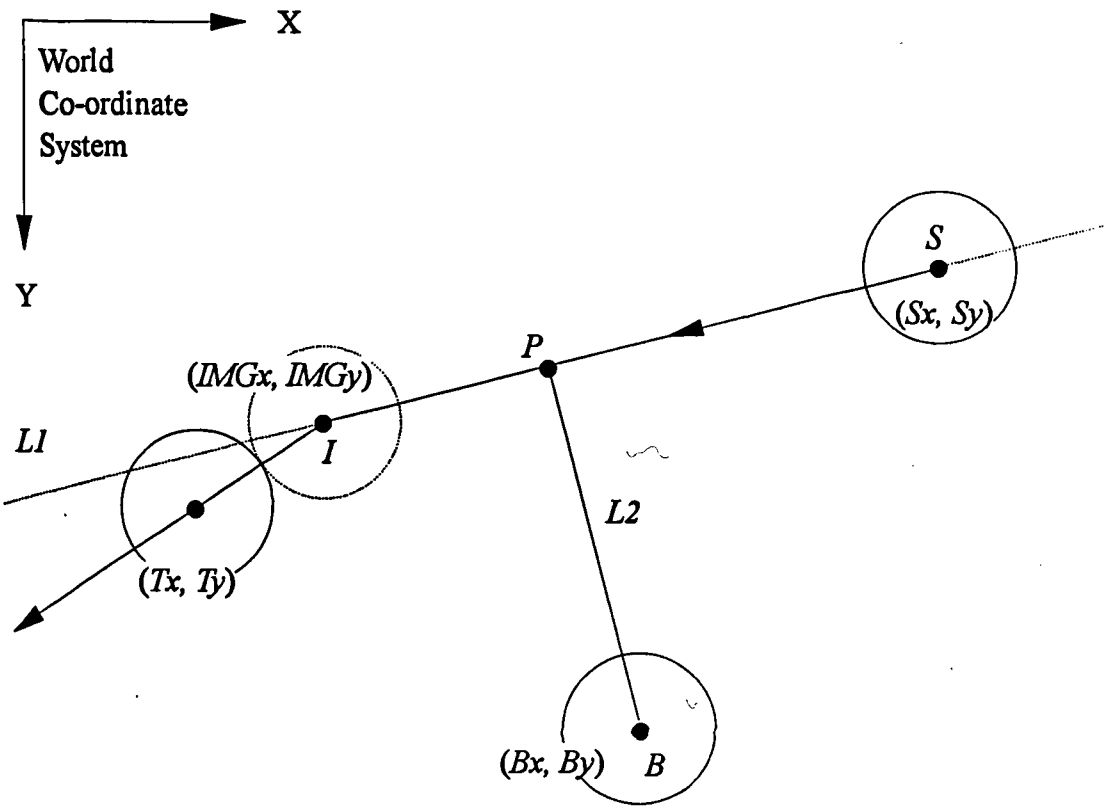
Figure 5.3 Necessary Condition For Potting a Traget Ball



D = Ball Diameter

For any ball on the snooker table, if its centre does not fall inside Rectangle A, the ball cannot obstruct the striking ball path.

Figure 5.4 Preliminary Test of Ball Path Obstruction

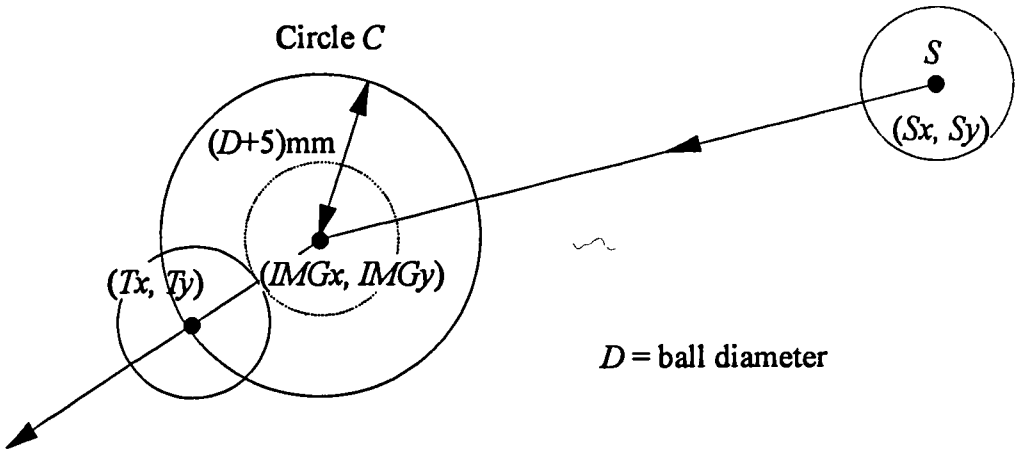


For any general ball centred at (*Bx, By*), there exists a point *P* on *L1* such that *SI* is perpendicular to *PB*.

The co-ordinates of *P* are (*Sx, Sy*) + *k* (*Dx, Dy*) where (*Dx, Dy*) = (*IMGx, IMGy*) - (*Sx, Sy*).

If $0 \leq k \leq 1$ AND $|PB| \leq (D+5)$ mm then the ball centred at (Bx, By) obstructs the striking ball path, where *D* = ball diameter.

Figure 5.5 Detection of Obstruction to Striking Ball Path



No ball other than the striking ball and the target ball can have its centre within Circle *C* if the striking ball is to make direct contact with the target ball.

Figure 5.6 Test to Ensure Direct Contact Between Striking Ball and Target Ball

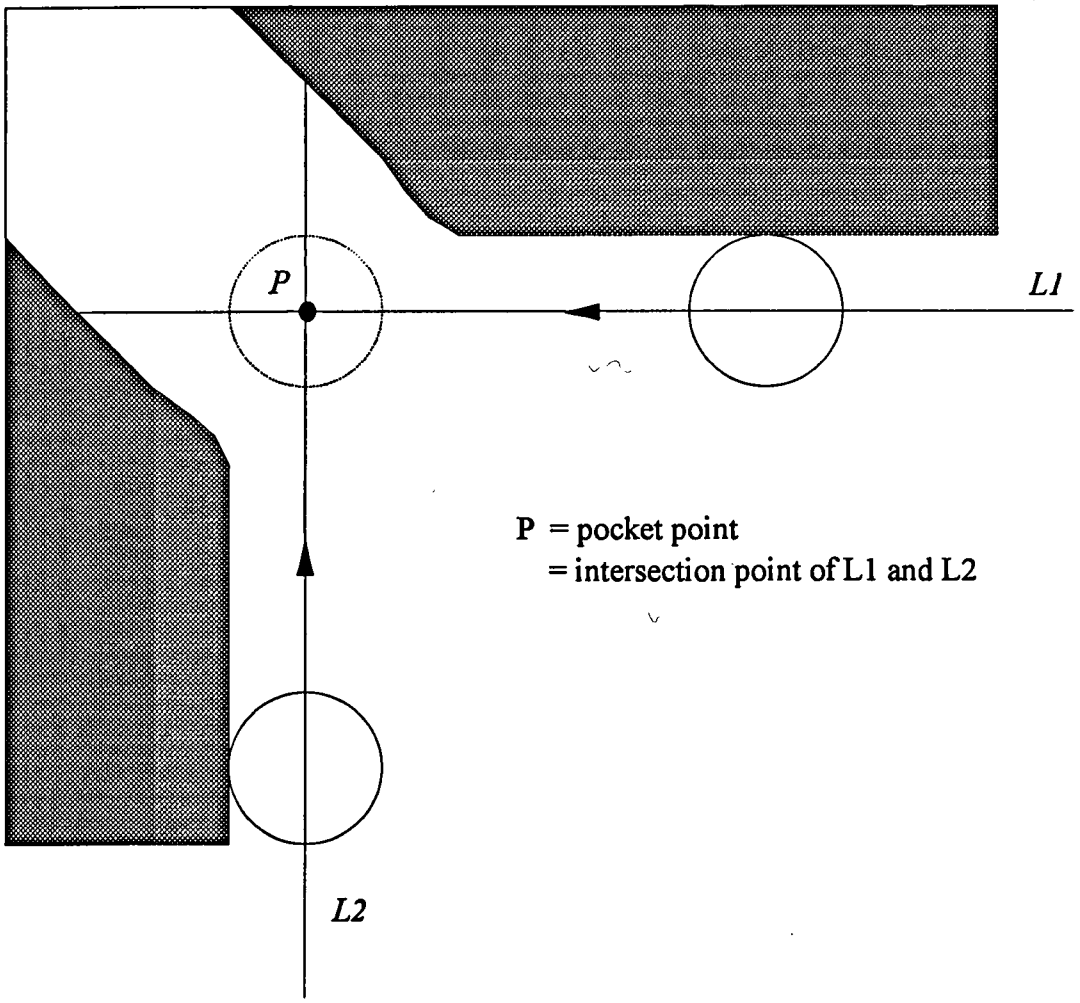


Figure 5.7 Definition of Pocket Point

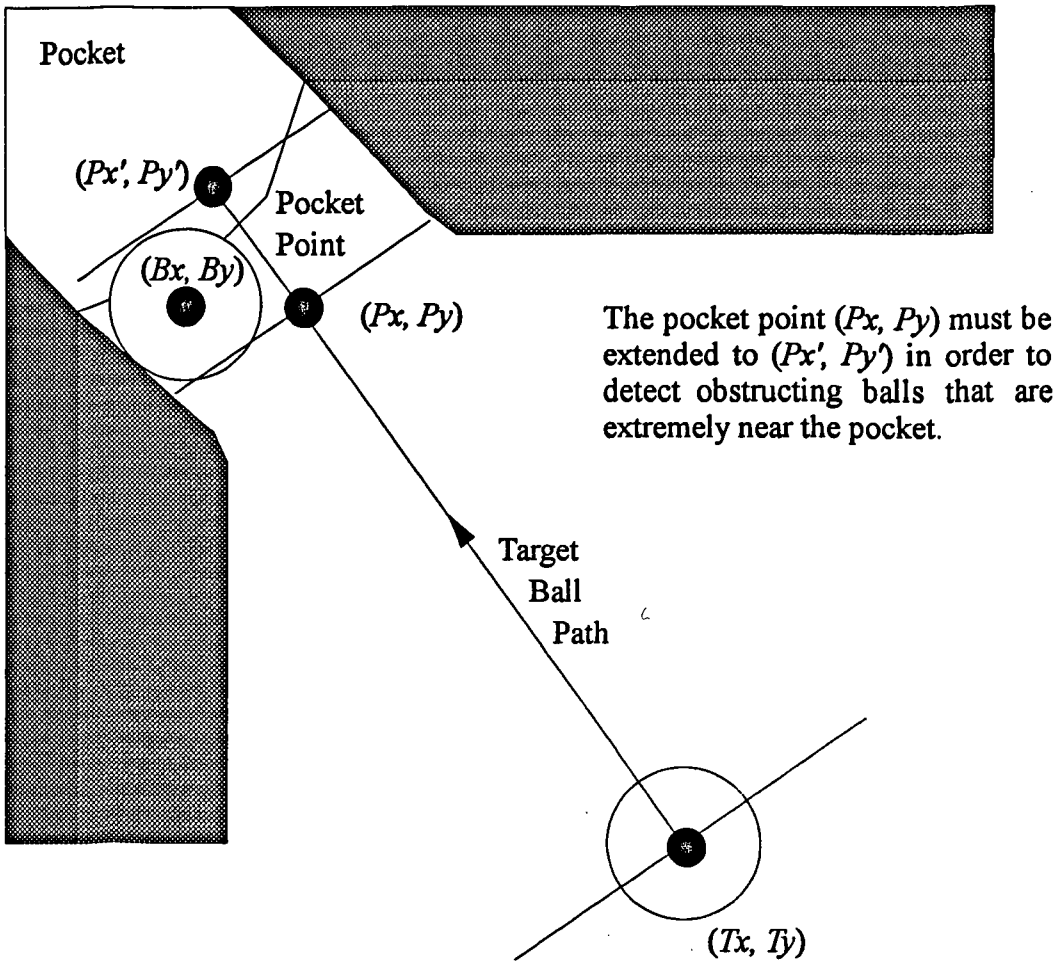
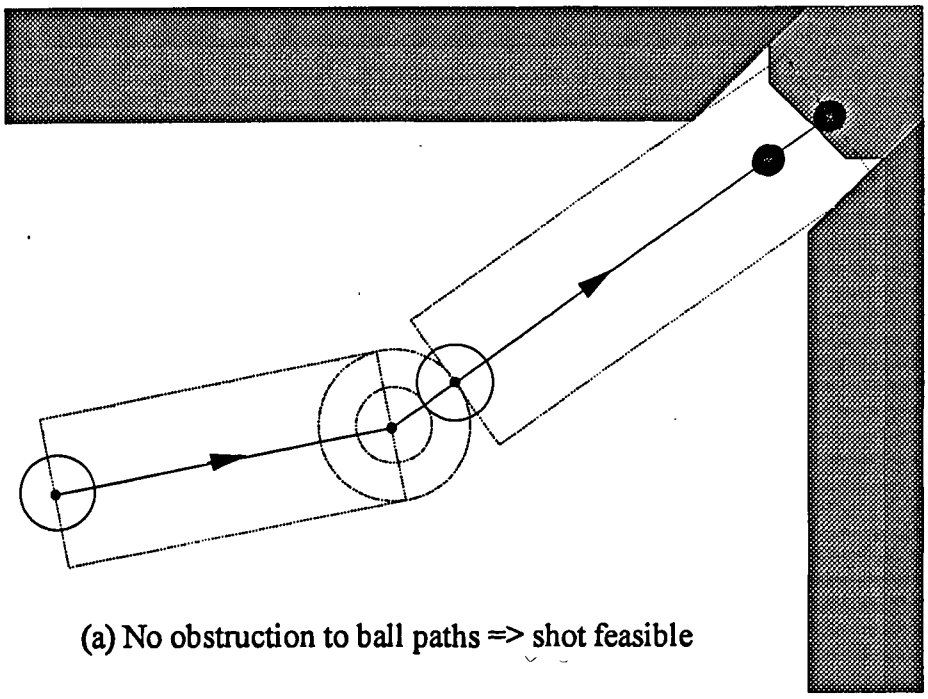
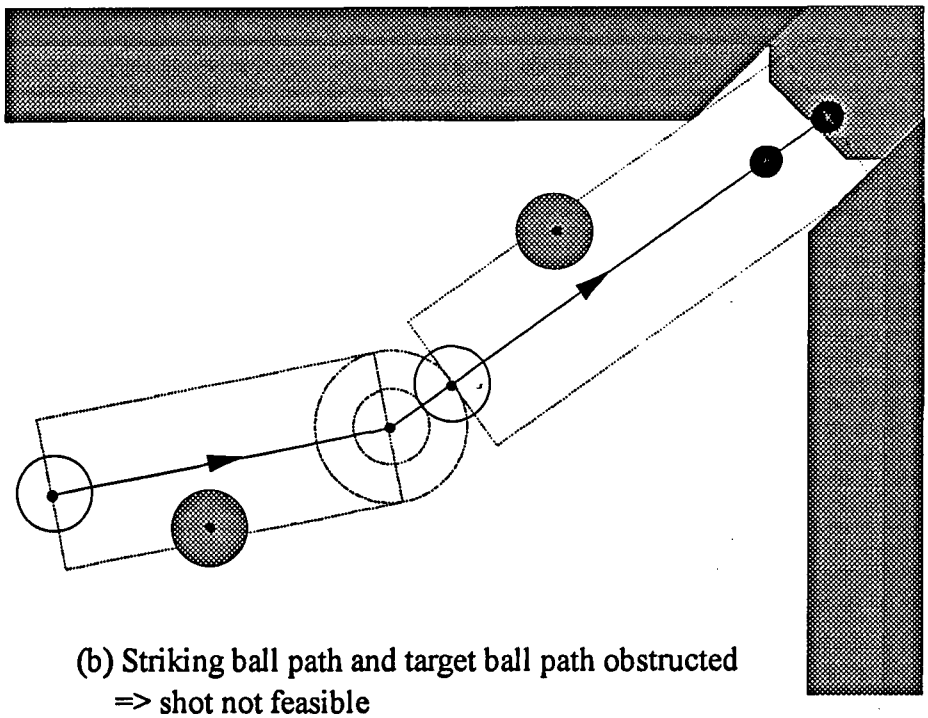


Figure 5.8 Detection of Obstructing Balls by Shifting Pocket Point Along Target Ball Path

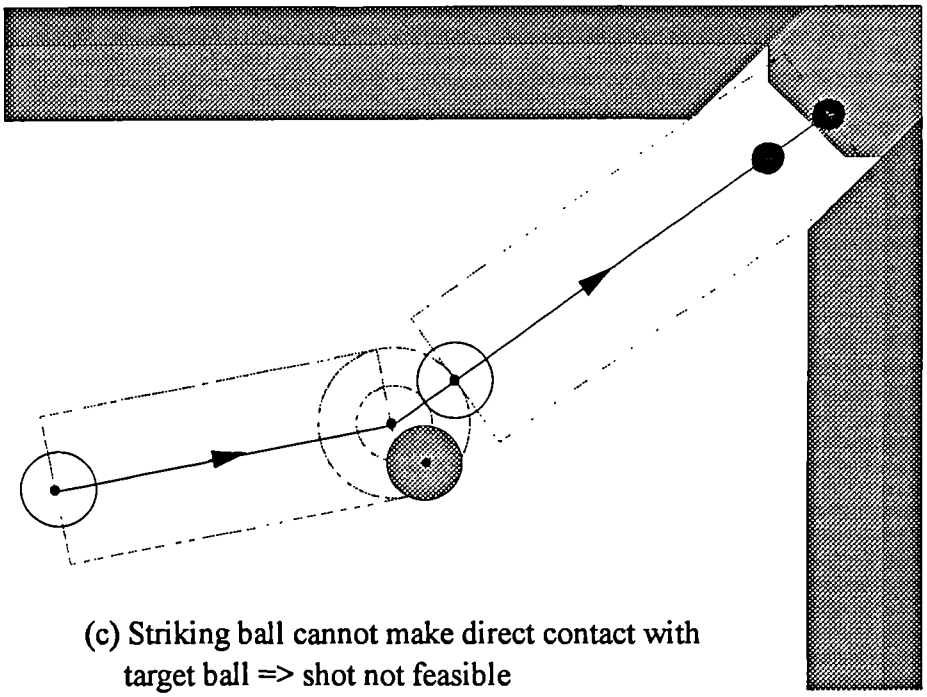


(a) No obstruction to ball paths => shot feasible

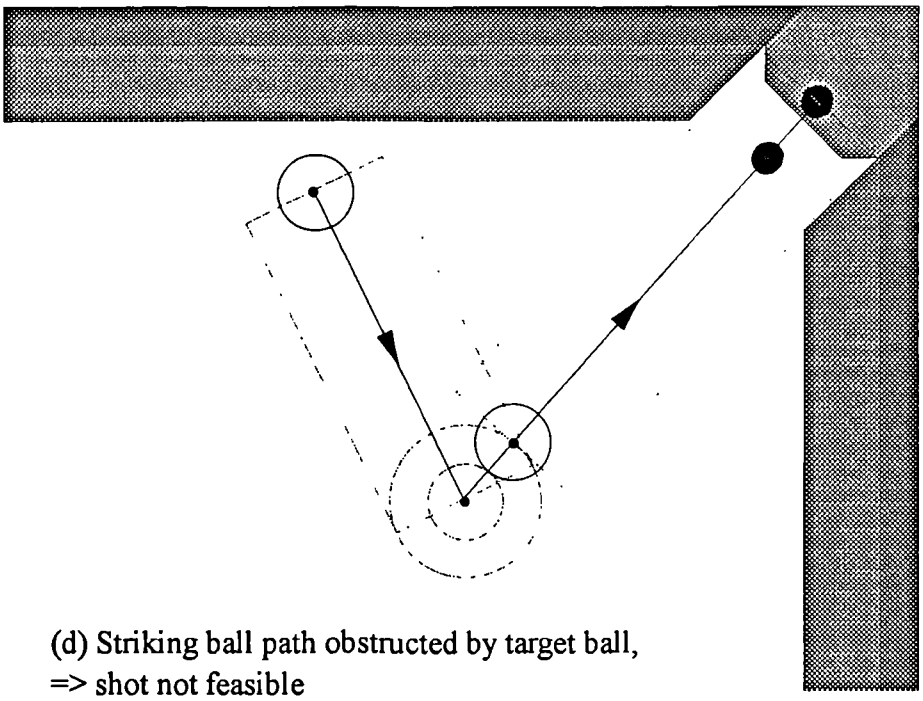


(b) Striking ball path and target ball path obstructed
=> shot not feasible

Figure 5.9 Sample Shots



(c) Striking ball cannot make direct contact with target ball => shot not feasible



(d) Striking ball path obstructed by target ball, => shot not feasible

Figure 5.9 Sample Shots

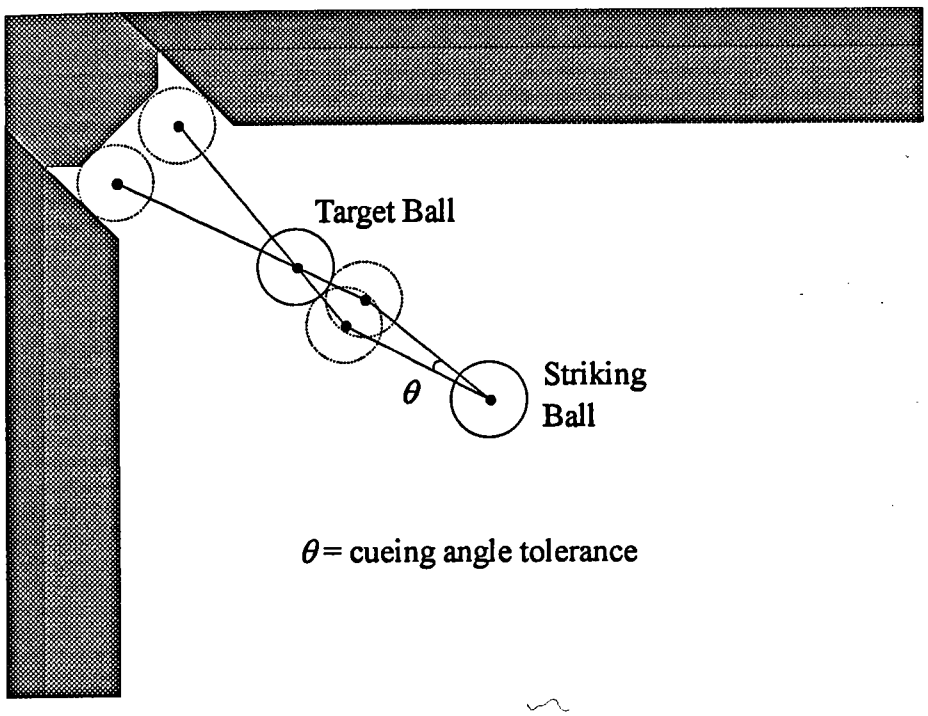
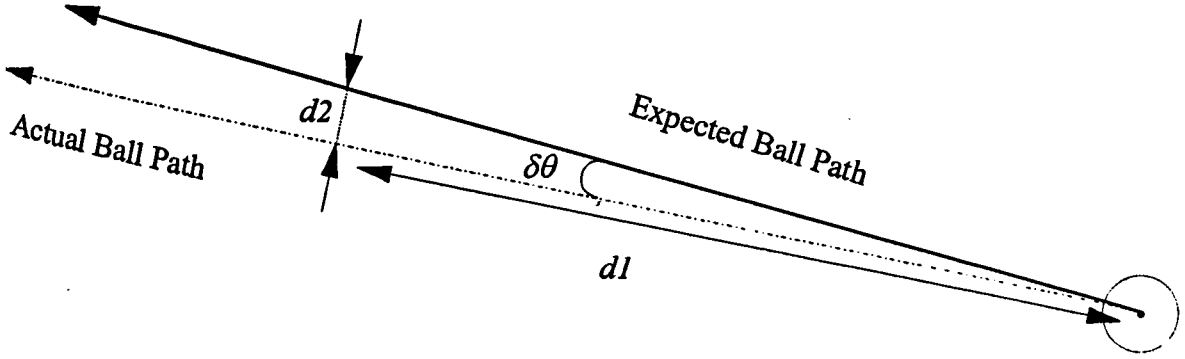


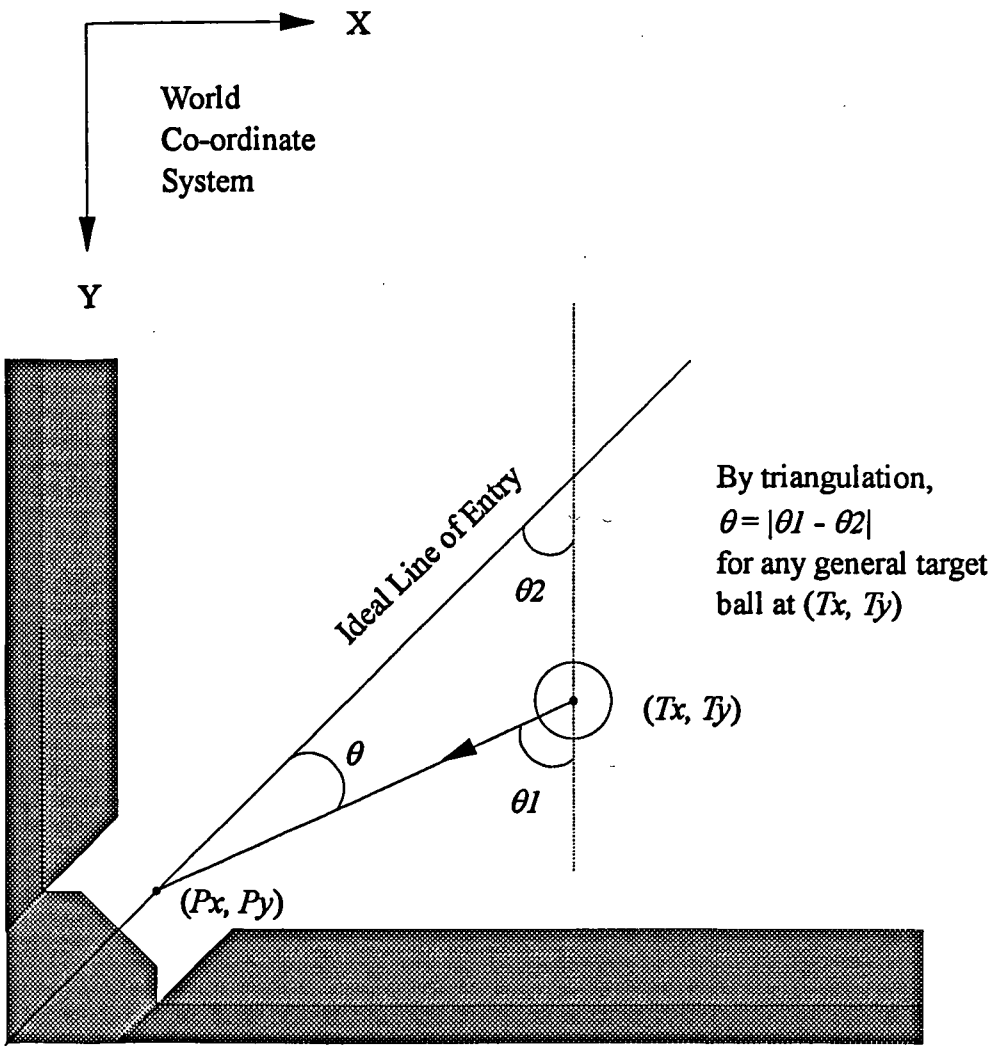
Figure 5.10 Example Showing Cueing Angle Tolerance



$\delta\theta$ = angular deviation from the expected path due to factors such as vision error or robot inaccuracy.

$d2$ = displacement from the expected path which is directly proportional to the distance travelled $d1$.

Figure 5.11 Relationship Between Displacement Error and Path Length



For each pocket, $\theta 2$, the angle between the ideal line of entry and the positive world Y-axis is known a priori.
In this example, $\theta 2 = 45°$.

Figure 5.12 Computation of Angular Deviation From The Ideal Line of Entry

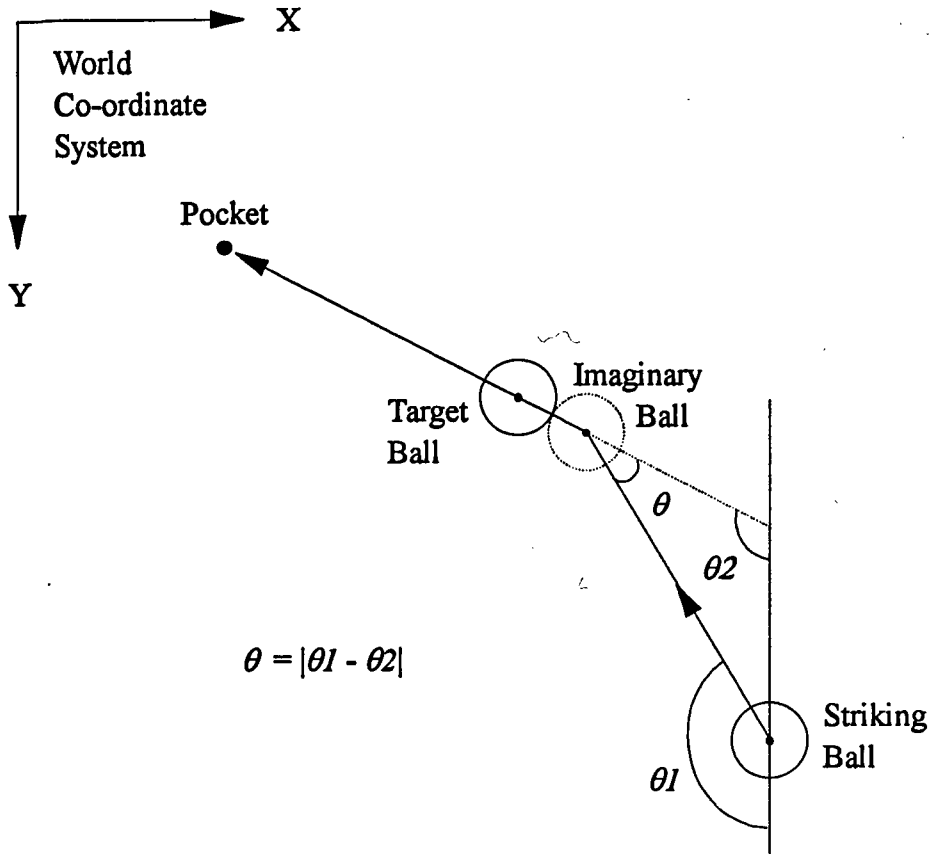
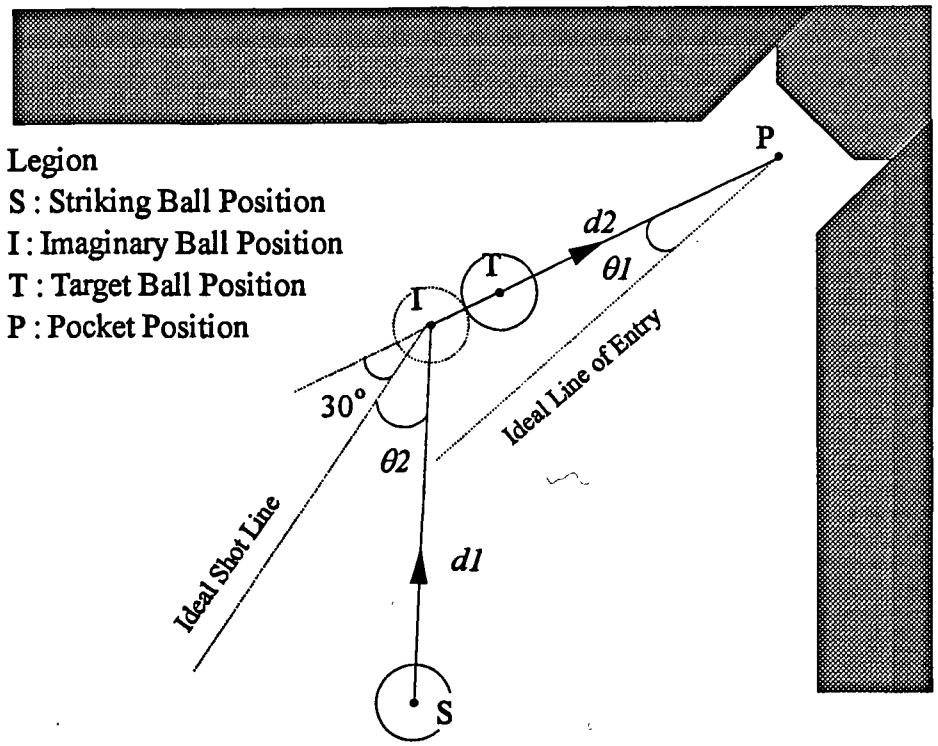


Figure 5.13 Angle Between Striking Ball and Target Ball Paths



Shot Difficulty Factors :

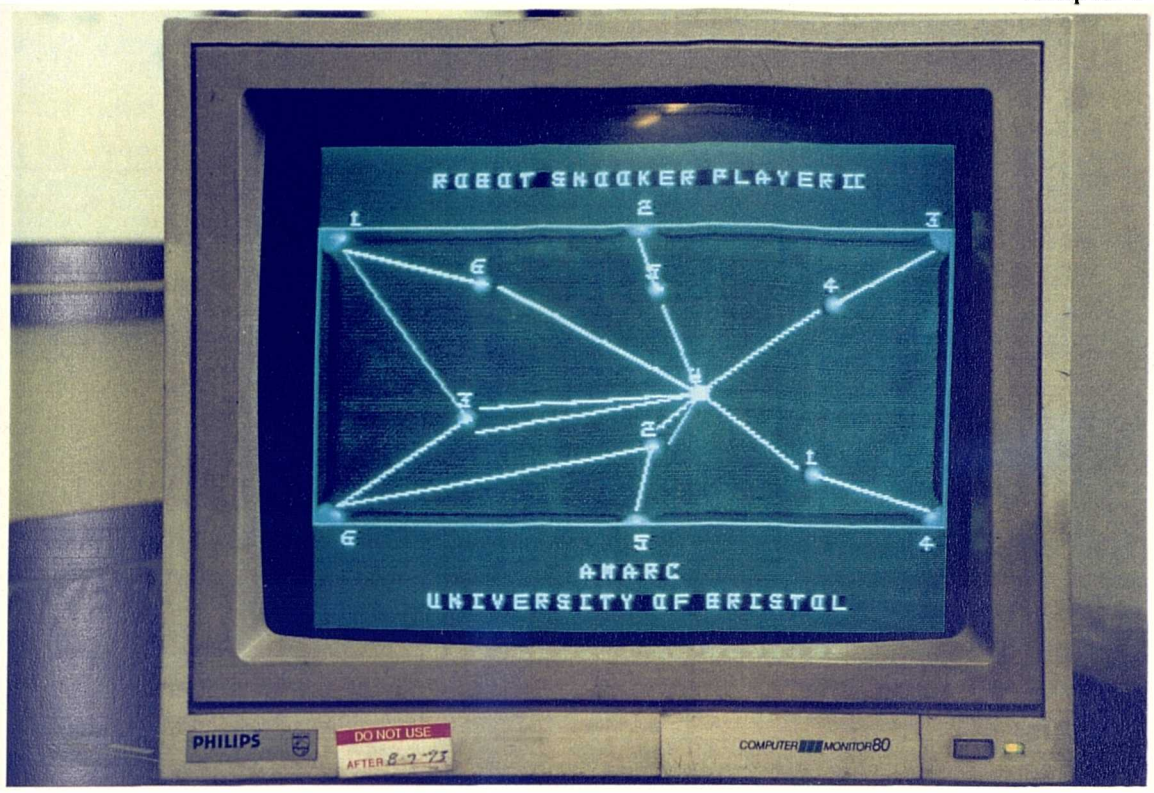
$SDF1 = d1/dmax$ $SDF2 = d2/dmax$

$SDF3 = \theta1/45°$ * $SDF1 = \theta2/60°$

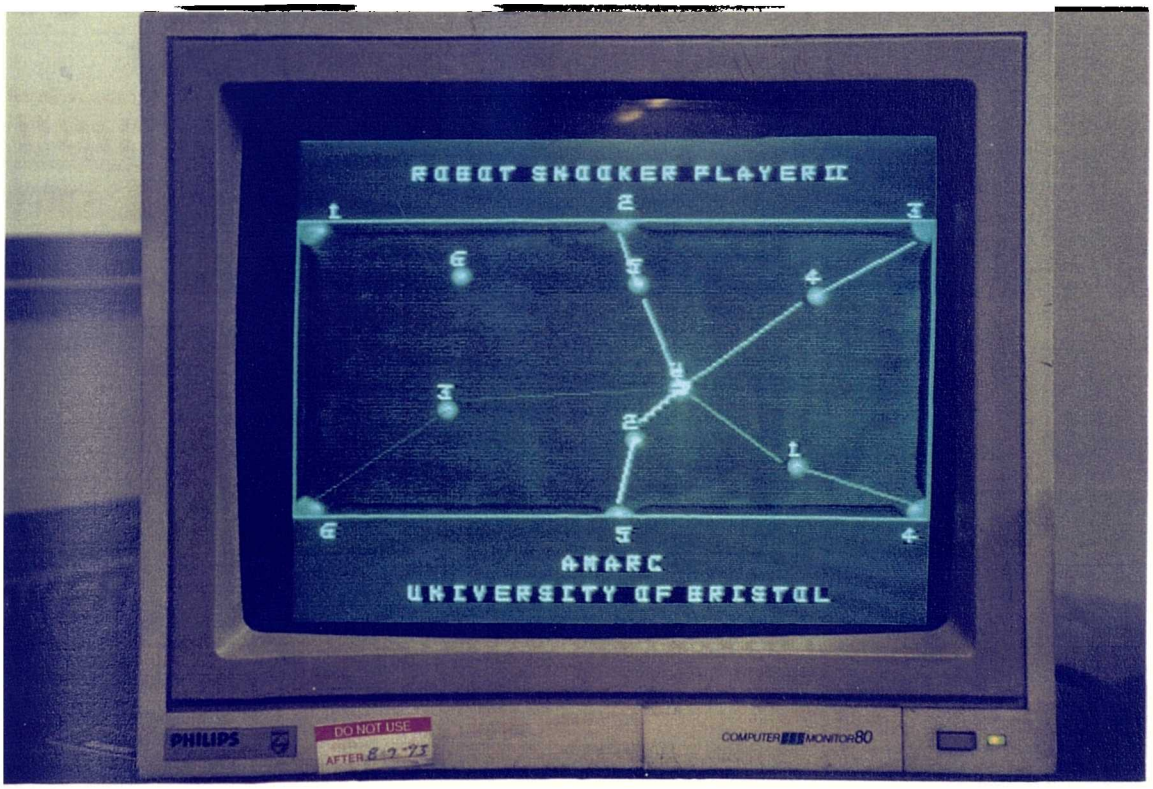
$dmax$ = length of the diagonal across the snooker table

* $SDF3 = \theta1/90°$ if a middle pocket is involved in the shot

Figure 5.14 Summary of Shot Difficulty Factors



(a) All feasible shots



(b) The Best 5 Shots
(Shot difficulty is inversely proportional to thickness of the shot line)

Figure 5.15 Sample Game Situation with 6 Red Balls and the Cue Ball

```
----- SHOT INFORMATION FOR RED BALL NUMBER 1 -----
RED Ball No.       Pocket No.       Shot Difficulty Index

    1                  4                1.740

----- SHOT INFORMATION FOR RED BALL NUMBER 2 -----
RED Ball No.       Pocket No.       Shot Difficulty Index

    2                  5                0.945
    2                  6                2.825

----- SHOT INFORMATION FOR RED BALL NUMBER 3 -----
RED Ball No.       Pocket No.       Shot Difficulty Index

    3                  6                2.035
    3                  1                2.729

----- SHOT INFORMATION FOR RED BALL NUMBER 4 -----
RED Ball No.       Pocket No.       Shot Difficulty Index

    4                  3                1.578

----- SHOT INFORMATION FOR RED BALL NUMBER 5 -----
RED Ball No.       Pocket No.       Shot Difficulty Index

    5                  2                1.029

----- SHOT INFORMATION FOR RED BALL NUMBER 6 -----
RED Ball No.       Pocket No.       Shot Difficulty Index

    6                  1                2.281
```

(a) Listing of all feasible shots

```
----- Shots of red balls ordered in ascending difficulty -----

Red Ball No.       Pocket No.       Shot Difficulty Index

    2                  5                0.945
    5                  2                1.029
    4                  3                1.578
    1                  4                1.740
    3                  6                2.035
    6                  1                2.281
    3                  1                2.729
    2                  6                2.825
```

(b) Sorted List of Shots in Ascending Order of Difficulty

Table 5.1 Shot Difficulty Index Listing Corresponding to the Sample Situation Shown in Figure 5.15

Chapter 6 Strategic Play : One-step Ahead Game Planning

In chapter 5, an outline of game playing without planning has been presented. Under a strategy with no planning, a shot with the highest chance of success (i.e. that associated with the lowest shot difficulty index) will be played regardless of the game situation after the shot is taken. Also there is no guidance as to what cue force is appropriate for the shot. The result is a haphazard game situation after the selected shot is played and no subsequent shot can be guaranteed. The availability of any subsequent shot is purely dependant on chance.

To enable the system to play snooker at a higher skill level, it is essential that the system is capable of anticipating the game situation after a particular shot is taken at a certain pneumatic cue force. A search can then be directed upon the anticipated game situation to assess the availability of any subsequent shots. The purpose of this sequence of actions is, in snooker terms, ' to gain position on the next ball ' which is similar to the thought process that goes through a human snooker player's mind. Given a number of shot possibilities, each of which could be taken with a range of cue forces, the shot that leads to a subsequent shot is always selected. In situations where there are more than one first and second shots combinations, the pair with the minimum combined shot difficulty index will be selected. Details of the planning strategy will be presented below.

6.1 Determination of the Minimal Cue Force for Potting a Ball

In chapter 3, the dynamic behaviour of snooker ball motion and collision was analysed in the forward sense : the behaviour of a stationary ball on receiving an impulsive blow, how the ball motion is opposed by friction and rolling resistance, and what happens when a moving ball collides with a stationary ball or a cushion were studied. Here, the reverse route has to be taken in order to calculate the minimum cue force required to pot a ball into a pocket, assuming that the pot is geometrically feasible with no obstructions to the striking and target ball paths. Bearing in mind that the pneumatic cue force is digitally controlled at 256 levels, it is important to determine the minimum cue force needed to pot a particular ball into a pocket to avoid under-hitting the target ball i.e. insufficient transfer of momentum to the target ball for it to reach the pocket.

The process of minimum cue force calculation can be broken down into 4 steps. Let (P_x, P_y), (T_x, T_y), (IMG_x, IMG_y), and (S_x, S_y) be the world co-ordinates of the pocket, target ball, striking ball position at impact with object ball, and the striking ball respectively, as shown in figure 6.1.

Step (1) Determination of initial target ball velocity such that it can just reach the pocket

Assuming that the target ball velocity at (Px, Py) equals zero, and knowing that the distance the target ball has to travel to reach the pocket equals d_1 (see figure 6.1), the initial target ball velocity, v_1, can be computed. Given the ball motion equations in figure 3.6, it can be derived that :

$$v_1 = \sqrt{\frac{d_1}{12/49 \mu g + 25/98 \dot{v}_r}}$$

where μ = coefficient of friction
$\dot{v}_r$ = resultant deceleration due to rolling resistance
g = gravitaional acceleration

Step (2) Determination of striking ball velocity immediately before impact with the target ball

The angle of impact θ is the angle between lines L_1 and L_2 (see figure 6.1), the target ball path after impact and the striking ball path before impact respectively. Given the coefficient of restitution of ball-to-ball impact, the velocity of the striking ball immediately before impact, v_2, can be computed using the equation in figure 3.10 :

$$v_2 = \frac{2v_1}{\cos\theta(1 + R)}$$

where R = coefficient of restitution of snooker ball impact

Step (3) Determination of initial cue ball velocity

Given that the striking ball velocity after travelling distance d_2 equals v_2, as computed at step (2), the initial striking ball velocity, v_3, can be determined. Since a function is already implemented to calculate the velocity of a ball after travelling a certain distance given an initial velocity, it can be utilised in a binary search for v_3, without the need to write a specific function to calculate v_3 directly. The function has the form :

$f(v_{init}, d, v_{final})$

where f is a BOOLEAN function
v_{init} and d are function input parameters
v_{final} is the output

Function f returns TRUE if a solution exists, in which case v_{final} is the ball velocity after travelling distance d. The function returns FALSE if the ball cannot travel far enough to cover the distance d with initial velocity v_{init}.

Note that any shot requiring a cue force greater than the maximum pneumatic cue force has to be rejected. This can be checked by assigning the maximum initial cue ball velocity corresponding to the pneumatic cue force level 256 (see Appendix C) v_{max} as the first function parameter v_{init} and calculate v_{final} , the striking ball's velocity after travelling distance d_2. A shot is rejected if function f returns FALSE, indicating that d_2 exceeds the maximum distance the ball can travel when struck with the maximum cue force or if v_{final} is less than v_2 (velocity of the striking ball at impact with the target ball), indicating that the pneumatic cue is incapable of delivering sufficient force to the striking ball such that after travelling distance d_2, the final ball velocity is less than that required. Otherwise a binary search utilising function f described above is initiated to determined v_3 to an accuracy of ±0.5 mm/s. Pseudo-code of the binary search can be seen in figure 6.2.

Step (4) Determination of initial cue ball velocity

Having determined the required minimum initial striking ball velocity v_3, the corresponding pneumatic cue force level can be found using a conversion function. Appendix C contains experiment results on the relationship between the pneumatic cue force levels and the initial ball velocity when struck by the cue. The relationship is modelled as linear which enables the initial ball velocity to be calculated given the force level and vice versa, given the ball velocity corresponding to the maximum pneumatic force level 256.

6.2 Prediction of the Striking Ball Path After Impact with the Target Ball

Suppose for a general shot, the minimum pneumatic cue force for potting a ball equals cf_{min}, it follows that the ball can be potted with a cue force cf such that $cf_{min} \leq cf \leq 256$, $cf \in$ integer. A mapping of pneumatic cue force to final striking ball position after a shot can be built up using the ball dynamics model. This involves converting the cue force cf into the corresponding initial striking ball velocity, which allows the velocity just before impact with the target ball to be calculated. Knowing the angle of impact, the striking ball velocity after impact can be calculated. Since the striking ball position at impact is known, and as are the locations of the snooker table cushions, the final position of the striking ball can be found by recursively checking if the striking ball hits a cushion before coming to rest. If the striking ball does hit a cushion, the reflected velocity will be calculated accordingly.

Note that the cue force / cue ball position map is not always complete. In fact the building of the map stops if any of the following conditions are true :

(i) the striking ball path is blocked by 1 or more balls

(ii) the striking ball is heading towards a pocket

Condition (i) ensures that the striking ball path is clear of obstacles since the dynamic model does not consider any secondary collision i.e. that involving a ball other than the striking or the target ball. Condition (ii) ensures that in potting the target ball, the striking ball will not drop into a pocket, which results in a foul shot. Figure 6.3 shows a sample of cue force / cue ball position map. The map of cue force to cue ball position is a crucial piece of information for shot planning. How this map is utilised in the 1-step ahead game planning algorithm will be seen in the next section.

6.3 Structure of a Snooker Game Tree

Leaving aside the precise calculation of the cue ball path after a shot is taken for the moment, playing a shot can be viewed as the transformation of a parent game state to a child game state. Each (target ball, pocket) shot combination can potentially lead to a collection of child game states since the shot can be played with a number of cue forces, resulting in different game situations. The difference between a parent state and its child state is that the target ball will be assumed to be potted while the cue ball assumes a new position on the table. All other balls are expected to remain in the same positions before and after the shot is played. In other words, each (target ball, pocket, cue force) shot

combination results in a unique child state. The cost of the transformation from parent to child state can be equated to the shot difficulty index associated with the (target ball, pocket) choice. Figure 6.4 is a diagrammatic representation of this concept.

To play snooker at a higher skill level, the system has to be able to anticipate the results of all actions which can be taken by the robot. This is made possible by the ability to control the pneumatic cue force together with the map between cue force and cue ball position, the building of which has been described earlier. The game situation after potting a ball into a pocket can thus be predicted before actually playing the shot. This predicted game situation forms the child state corresponding to that particular shot played at a certain cue force. In each child state, provided at least 1 ball can be potted into a pocket, there exists a best shot associated with the minimum shot difficulty index. This can be seen as shot selection without planning directed upon the child state. The shot difficulty index of the best shot can be considered as the cost of the child state.

Figure 6.5 shows the structure of a simplified snooker game tree. From the parent state, child state i can be reached, incurring a cost C_{1i} that equals the difficulty index of the shot selected from the parent state. Playing the shot transforms the game situation into child state i. Let the cost associated with child state i equals C_{2i}. The total cost to traverse branch i therefore equals $(C_{1i} + C_{2i})$. The objective of the 1-step ahead game planning algorithm is to find branch i of the game tree such that the total cost is minimised, thereby maximising the chance of success.

6.4 1-Step Ahead Game Planning Methodology

To understand the 1-step ahead planning methodology, it is necessary to take an in-depth view of the game tree in figure 6.5. A 1-step ahead game plan involves the determination of an initial shot followed by another shot i.e. a sequence of 2 shots where the combined shot difficulty is minimal. Thus potting a red ball followed by a colour ball is one possible sequence while the consecutive potting of two red balls is not. The search for the best shot sequence involves walking through the game tree and keeping a record of the shot sequence associated with the minimal combined shot difficulty index. This search can be broken down into two distinct processes : (1) the generation of child states from a parent state following the potting of a ball into a pocket and (2) the scanning of child states for any possible subsequent shot.

In the following, the potting of a red ball followed by a colour ball will be taken as an example. For other potting sequence, such as potting a colour and then a red, the same method is applicable.

6.4.1 Generation of Child States from the Parent State

Figure 6.6 shows a general parent state where there are n red balls, the yellow, green, brown, blue, pink, black and the cue ball on the table. Each red ball in the parent state is checked against every pocket to determine whether a shot is feasible. For each feasible (red ball, pocket) combination, a cue force / cue ball position map is generated using the ball dynamics model (see section 6.2 above). Each entry, excluding ' null ' entries, in that map allows the game situation after potting the ball to be created, leading to a unique child state, as shown in figure 6.6.

Note that all child states that are descendants of the same (red ball, pocket) combination, e.g. red ball number 1 into pocket 4, are classified as members of the same family and are identical in terms of ball positions except for that of the cue ball. The transformation cost from the parent state to child states of the same family are identical.

6.4.2 Scanning of Child States for the Best Subsequent Shot

According to the rules of snooker, one of the six colour balls can be potted after potting a red ball. Hence for each child state, each colour ball is checked against every pocket to determine whether it can be potted. For each feasible (colour ball, pocket) combination, the shot difficulty index is calculated. Suppose the shot difficulty index of a (colour ball, pocket) combination equals SDI_q, and the transformation cost from the parent state to the current child state equals SDI_p. The combined shot difficulty is thus $(SDI_p + SDI_q)$. If the value of $(SDI_p + SDI_q)$ is less than the current minimal, the latest best first and second shot sequence has been found. The relevant shot data will be recorded and $(SDI_p + SDI_q)$ becomes the new minimal combined shot difficulty. The relevant shot data includes the number of the red ball to be potted, the number of the pocket, and the cue force at which the shot is to be played at... This search mechanism for finding feasible shots in a child state is summarised in figure 6.7.

6.5 Tree Search Implementation

Figure 6.8 shows an example of a partial snooker game tree. Each intermediate node in each level of the tree can be the root of a subtree. Suppose there are 10 feasible (red ball, pocket) combinations within the parent state, and that each shot can be played at 50 different force levels, there will be a total of 500 child states each of which is the root of a subtree. This is a conservative estimation but nevertheless illustrates the size and complexity of a snooker game tree which is only 2-ply deep.

For optimal performance, the snooker game tree has to be searched in an orderly and efficient manner. Searching the tree ' breadth-first ' is ruled out because of the excessive associated storage requirement. A dynamic data structure would have to be maintained to store each cue force / cue ball position map associated with each (red ball, pocket) combination. This map is the key information that allows a family of child states to be generated from a (red ball, pocket) combination. Since each map is a 256 × 2 array of real numbers, the amount of memory required to store all the mappings will be highly excessive.

Searching the snooker game tree in a 'depth-first' manner proves more economical as only the map associated with the current (red ball, pocket) combination needs to be maintained. When the subtree emerging from a (red ball, pocket) combination is exhaustively searched, the cue force / cue ball position map is no longer needed. On reaching the subsequent (red ball, pocket) combination, a new map will be generated to replace the old map.

Prior to the beginning of the tree search, a data structure for recording shot data has to be initialised. This is required to keep track of the game plan consisting of a first and second shot. Basically the data structure 'shotseq' is a 10 × 2 array, as shown in figure 6.9, which stores the ball identification number, pocket number, shot difficulty index, and cue ball position. For the first shot only, the selected cue force is also recorded. Another 2-dimensional array ' bestmap ' records the cue force / cue ball position map associated with the current best plan. The information carried by the array 'shotseq' and 'bestmap' allows the game plan to be displayed on the CRT display, superimposed over the digitised image of the snooker table, see figure 6.10. Initially, the array elements corresponding to the shot difficulty index ' shotseq [1,7] ' and ' shotseq [2,7] ' are assigned an artificially high value of 1000, and the Boolean flags ' shotseq [1,1] ' and ' shotseq [2,1] ' are set as FALSE; which indicate that no shot has been found initially.

Referring back to figure 6.8, it can be seen that the pocket into which a colour ball can be potted is a terminal node of the game tree. On reaching a terminal node, the cost to traverse that particular path of the tree equals the combined shot difficulty of potting a red ball followed by potting a colour ball. Using the example tree in figure 6.8, the cost to traverse Path A equals the sum of the shot difficulty index of potting red ball number 1 into pocket number 4 in the parent state and the shot difficulty index of potting the yellow ball into pocket number 1 in child state u, which equals $(p + r)$. Now suppose $(p + r)$ is less than the current value of ' shotseq[1,7] + shotseq[2,7] ', the shot data corresponding to Path A will be recorded into the array 'shotseq' to replace the previous best plan. Note that all the information to be recorded into 'shotseq' are clearly defined along the path. Similarly, on reaching the termination of Path B in figure 6.8, if $(q + s)$ is

less than current best combined shot difficulty, then potting red ball number 2 into pocket number 6 at cue force cf_n followed by potting the blue ball into pocket 5 becomes the latest best plan. Otherwise no change is made to the plan as recorded in the array ' shotseq '.

The 1-step ahead planning algorithm always try to find the best plan of a sequence of 2 shots. However, it is possible that no feasible second shot exists in all the child states, and therefore no 1-step ahead plan exists. To ensure that the machine has a shot to play in such a situation, the best possible shot in the parent state has to be identified. Since a child state is the result of transformation of a parent state through the process of potting a ball, the associated shot difficulty index, also known as the transformation cost, is always computed before reaching a child state. Thus a set of shot data corresponding to the current first shot always exists on reaching a child state. Referring back to figure 6.7, if a ball cannot be potted into a pocket, the array 'shotseq' is accessed and checked. If a plan already exists, the flags ' shotseq [1,1] ' and ' shotseq [2,1] ' will both be TRUE, and no action is taken. On the other hand, if 'shotseq [2,1]' is FALSE, the system has at most located 1 shot. If the difficulty of the current shot identified in the parent state is less than the array element ' shotseq [1,7] ' (shot difficulty index of the current best shot), the latest best first shot has been found and the associated shot data will be recorded.

6.6 Game Plan Scheduler

In the above section, potting a red ball followed by a colour ball has been used as an example of shot planning. During an actual game of snooker, there are other playing sequences. For example, any one of the 6 colour balls i.e. yellow, green, brown, blue, pink, and black can be potted after potting a red. However, playing a red ball after potting a red is a foul shot. By checking the number of red ball on the table and a Boolean flag signal which indicates whether a red ball has just been potted, the correct potting sequence can be identified, thus avoiding the possibility of committing a foul shot by playing the wrong ball.

The key to finding these sequences is to realise that when a player starts playing, either there is 1 or more red balls on the table or there is none. Suppose a player successfully pots every ball until none is left, then all the possible shot sequences of 2 shots must be present in the clearing of all the balls. Such a total clearance is graphically represented in figure 6.11. As can be seen in the figure, there really are only 4 different sequences of shots :

***SS-I* (*S*hot *S*equence *I*) Red-Colour** : any 1 of the available red balls can be potted followed by the potting of any 1 of the 6 colour balls. The red ball, if potted, is removed from play.

***SS-II* Colour-Red** : any 1 of the 6 colour balls can be potted followed by the potting of any 1 of the available red balls. The colour ball, if potted, is replaced on its spot on the table.

***SS-III* Colour-Yellow** : any 1 of the 6 colour balls can be potted after which only the yellow ball can be played. The first colour ball, if potted, is replaced on the table.

***SS-IV* Clear-Colours** : the colour ball of the lowest points scoring value has to be potted followed by the ball of the next lowest value. The 6 colour balls in order of ascending values are listed below :

Yellow (2 points), Green (3), Brown (4), Blue (5), Pink (6), Black (7)

The colour ball of the lowest value is removed from play if potted. For example, if only the last 4 colour ball i.e. Brown, Blue, Pink, and Black are left to be potted, the current ball to play is the Brown followed by the Blue.

Note that the beginning of a series of shots has to begin with either *SS-I* or *SS-IV*. A red ball has to be potted before arriving at *SS-II* and *SS-III* if the rules on potting order is not to be violated. Indeed *SS-I* to *SS-IV* encompass all legitimate sequences of 2 consecutive shots in a game of snooker. A potted red ball is never replaced on the table. Whether a potted colour ball is replaced on the table depends on the shot sequence. In the generation of child states from a parent state, this piece of information must be accurately reflected.

Having identified the different shot sequences, a reliable method of determining which is the correct sequence to play is required. The game planning function is always invoked after a fresh image of the snooker table is digitised and the balls on the table identified. The global variable 'Nred' records the total number of red balls on the snooker table. A flag 'Prev_Is_Red' indicates whether a red ball has just been potted. 'Prev_Is_Red' is always initialised as FALSE, and is set as TRUE only if a red ball has just been potted. The flow diagram in figure 6.12 shows how the correct playing sequence is identified according to the values of 'Nred' and 'Prev_Is_Red'. Each of the 4 possible playing sequence are basically identical in terms of the game tree structure. However, because of some subtle differences in the construction of child states from parent states, 4 game

planning functions, corresponding to the 4 shot sequences, are designed and implemented instead of a single general game planning function.

6.7 Dealing With Game Situations In Which No Ball Can Be Potted

For the robot snooker player, the ideal game plan is to be able to devise a plan of 2 consecutive shots using the 1-step ahead planning algorithm. In reality, however, this may not be achievable. In such cases, alternative course of action must be considered.

To find a plan of 2 consecutive shot is of the highest priority. In the ideal situation, a (target ball, pocket, cue force) combination would have been determined by the game planning functions and upon playing the shot the cue ball would arrive at a position where another ball can be potted as predicted. If no such plan can be found in the current game situation, the best single shot will be played. How the current game situation is systematically searched for the best sequence of 2 shots or a single shot has already been dealt with in chapter 6.5. Here a strategy to deal with game situations where no ball can be potted is devised.

In a general game situation, if no ball can be potted into any pocket, the robot snooker player is effectively force to play defensively. Such a game situation will be referred to as a No-Pot Situation. Furthermore, a No-Pot Situation falls into 1 of 2 categories : Non-Snookered or Snookered.

In a Non-Snookered No-Pot Situation, the cue ball can directly hit a least 1 of the legitimate target balls. But the target ball cannot be potted either because the required cue ball path to pot the ball is obstructed by another ball or the target ball path to enter a pocket is blocked by another ball. Examples of Non-Snookered No-Pot Situations can be found in figure 6.13 (a).

In a Snookered No-Pot Situation, the robot snooker player cannot hit any of the legitimate balls directly because of blockages. For example, if the current shot sequence to play is Red-Colour and the cue ball cannot hit any of the red balls directly, the state of the game is a Snookered No-Pot Situation. In order to play the blocked target ball, 1 or more of the table cushions can be used to redirect the cue ball towards the target ball. Admittedly sometimes a snookered ball can be potted in such indirect shots. However this type of trick shots are often flukes and are rarely played by professionals intentionally because of the high risk of failure, which is why the robot snooker player is not programmed to do so. Figure 6.13 (b) contains examples of Snookered No-Pot Situations.

A No-Pot Situation is identified if after invoking 1 of the 4 game sequence planning functions, no shot is recorded in the game plan data structure ' shotseq '. When this happens, the system has to adopt a strategy of damage limitation. Instead of trying to pot a target ball, the aim becomes one of making it difficult for the opponent to pot a ball.

Being in a Snookered No-Pot Situation is more dangerous than in a Non-Snookered No-Pot Situation because using a cushion to rebound the cue ball at a target ball is always more risky than aiming to hit a target ball directly. Therefore, when in a No-Pot Situation, the system always tries to hit a ball directly. Only when the system fails to find a target ball which the cue ball can hit directly would it concede that all balls are snookered.

6.7.1 Dealing With a Non-Snookered No-Pot Situation

A No-Pot Situation is always treated as though it is a Non-Snookered one first. A search is then conducted to find the least risky ball to hit directly. The level of risk is based on the distance the cue ball has to travel before making contact with the chosen target ball. The objective is to hit the cue ball with a cue force such that the cue ball has just enough momentum to reach the target ball. The reasoning for such a shot being a safety shot is that if a ball cannot be potted, it should remain so with the cue ball very near it. Also if more than one ball can be directly hit, the one nearest the cue ball is always chosen in order to minimise the risk of missing the ball.

The search involves scanning each target ball that the cue ball can legitimately hit directly. Any visit to the snooker table falls into 1 of 3 categories : (1) 1 of the available red balls can be played, (2) any 1 of the 6 colour balls can be played, or (3) the lowest point scoring colour ball is to be played. What is needed therefore is a function that can determine if a ball can be directly hit by the cue ball. The function can then be applied to every ball that can be played.

A data structure similar to that for recording the shot plan is required. In this instance, a 1-dimensional array of 6 elements is sufficient to record the relevant data on a safety shot, see figure 6.14. For each target ball, the locus of all possible cue ball centres at impact with the target ball is a circle centred at the target ball, and the diameter of the locus equals the snooker ball diameter. This locus of infinite cue ball centres at impact is then approximated by 90 points at 4° intervals on the locus, as in figure 6.15.

Each of these 90 interval points is projected towards the cue ball and a check on path obstructions is carried out. The cue ball can directly hit the target ball through a

particular interval point on the locus if and only if no obstructing balls exists. Note that the target ball itself counts as an obstructing ball and hence all interval points behind the target ball relative to the cue ball will always fail the test for obstructions.

If the cue ball can hit the target ball directly through more than 1 interval points on the locus, the point nearest to the cue ball will be selected such that the most direct hit is always preferred. Thus for each valid target ball, the interval points approximating the locus of cue ball centre at impact are determined. For each interval point, if no obstruction exists between the point and the cue ball and the distance between them are less than that recorded in the array 'safety_shot', a new safety shot has been found and the relevant data will be recorded.

6.7.2 Dealing With a Snookered No-Pot Situation

In a situation where no valid target ball can be directly hit by the cue ball, the cue ball has to hit a target ball via an indirect path. In professional play, this can be achieved by the application of side to the cue ball resulting in drastic swerving of the cue ball, thus bypassing any obstructing balls. The robot snooker player is incapable of replicating this type of shots for a number of reasons. The risk involved is much higher than playing plain shots, cueing has to be extremely precise, and a cue force far exceeding the pneumatic cue's power output is likely to be required. As a result, the robot snooker player is programmed to deal with a snookered situation in a more conventional way.

A less risky but no less effective way to hit a snookered target ball is to direct the cue ball at 1 of the 4 cushions. Before moving on, a few assumptions have to be made to help speed up the search for a solution to the snookered game situation :

Assumption (1) : When a ball hits a cushion, the reflected angle is equal to the incident angle.

Assumption (2) : No energy is lost through impact between a moving ball and a cushion.

The above assumptions may appear as too simplistic and indeed these assumptions were not made when planning shots. However, determination of the shot angle to pot a ball does demand the highest possible precision which is why the coefficient of ball-to-cushion restitution and momentum loss has to be taken into account. It has to be stressed that before reaching a Snookered No-Pot Game Situation, the system has already searched the current game situation for a 1-step ahead game plan, followed by a search for the best single shot, then followed by the searched for a safety shot, all of which must have failed. Several minutes of computation time is likely to have elapsed

when a snookered situation is finally identified, which is why it is desirable to solve the problem using heuristics rather than exact calculations.

The first step to solving the problem is to realise that a cushion can be visualised as a mirror such that for a ball at position (x, y), there exists a reflected ball position (x', y') as shown in figure 6.16. The position of an obstructed target ball can thus be mirrored off each of the 4 table cushions. This is easily achieved because the positions of the cushions in world co-ordinates are known, which allows the cushions to be modelled as lines parallel to the X or Y axes as shown in figure 6.17 which also shows how the mirrored ball positions are computed given a general ball at world co-ordinates (x, y).

From here onwards, the mirrored ball position off the top cushion will be taken as an example. Let the mirrored ball position off the top cushion be (topX, topY). The line joining (topX, topY) and the cue ball at (cueX, cueY) will cut across the top cushion as shown in figure 6.18. The intersection point can be determined using the same function used in the tracing of the cue ball path in chapter 6.2 earlier. Suppose the co-odinates of this intersection point equals (cutX, cutY), it follows under Assumption (1) above that the cue ball, if directed at (cutX, cutY), will hit the originally obstruction target ball at (x, y). This is supported by the geometric verification shown in figure 6.19. Provided that no obstruction exists between (cueX, cueY) and (cutX, cutY) and between (cutX, cutY) and (x, y), and that (cutX, cutY) is not near a pocket, the cue ball can be played towards (cutX, cutY) to hit the snookered target ball at (x, y).

Similar to the playing of a safety shot, the minimum cue force is to be use in such a shot so that the cue ball is delivered just enough momentum to reach the target ball. To reach the target ball, the cue ball has to travel from (cueX, cueY) to (x, y) via (cutX, cutY), thus allowing the total distance traversed to be calculated. Under Assumption (2) above, no momentum is loss through the cushion impact. Hence given that the cue ball velocity at (x, y) equals 0 and the distance travelled, the initial cue ball velocity can be determined which can then be converted into the equivalent pneumatic cue force.

The above procedure is carried out on each of the 4 mirrored ball positions for every obstructed target ball. Again, it is possible to have more than 1 ball-cushion solution. Admittedly, the projected cue ball path under Assumption (1) and (2) will not match the actual cue ball path exactly knowing that the reflected angle is always greater than the incident angle. To minimise the amount of path deviation, the shot requiring minimal cue ball travel before impact will be selected if more than 1 solutions exists.

6.8 Conclusions

A systematic set of rules have been formulated to determine the best course of action given a random game situation. The most challenging task is for the robot snooker system to similate the human skill of game planning. This has been achieved by utilising the understanding of ball dynamics gained in chapter 3 so that the possible path taken by the cue ball after hitting a ball can be predicted. This allows a mapping between the pneumatic cue force used and the final finishing position of the cue ball to be constructed. This in turn facilitates the building of a game tree in which the current game situation is the parent node. By strategic choice of target ball, pocket, and the cue force used to pot it, a favourable game situation will be arrived at.

In actual play, however, there may be situations where the system fails to arrive at a game plan. Indeed there may be times when no balls can be potted at all. To enhance the robustness of the robot snooker player, it must be able to deal with these situations. The following are the possible choices of action that can be taken by the robot snooker player, arranged in the descending order of priority :

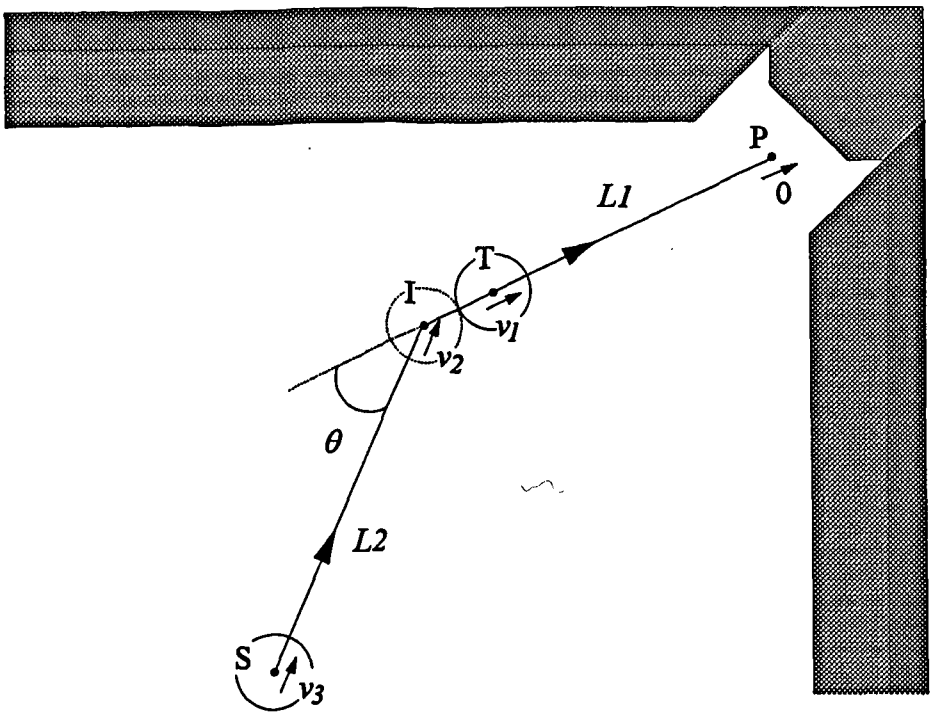
(1) plan a sequence of 2 shots (highest priority)

(2) play the lowest difficulty shot

(3) play a safety shot

(4) play at a cushion to get out of a snooker (lowest priority)

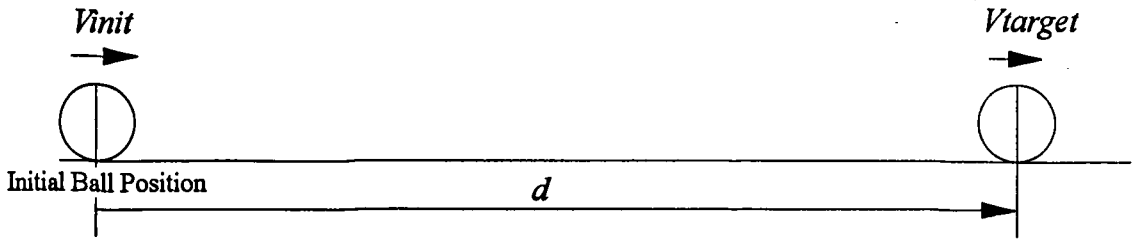
If none of the above actions can be taken, the system is stuck and will have to concede a shot. In actual snooker play, this kind of situation in which the system runs out of options has never occurred.



Legion

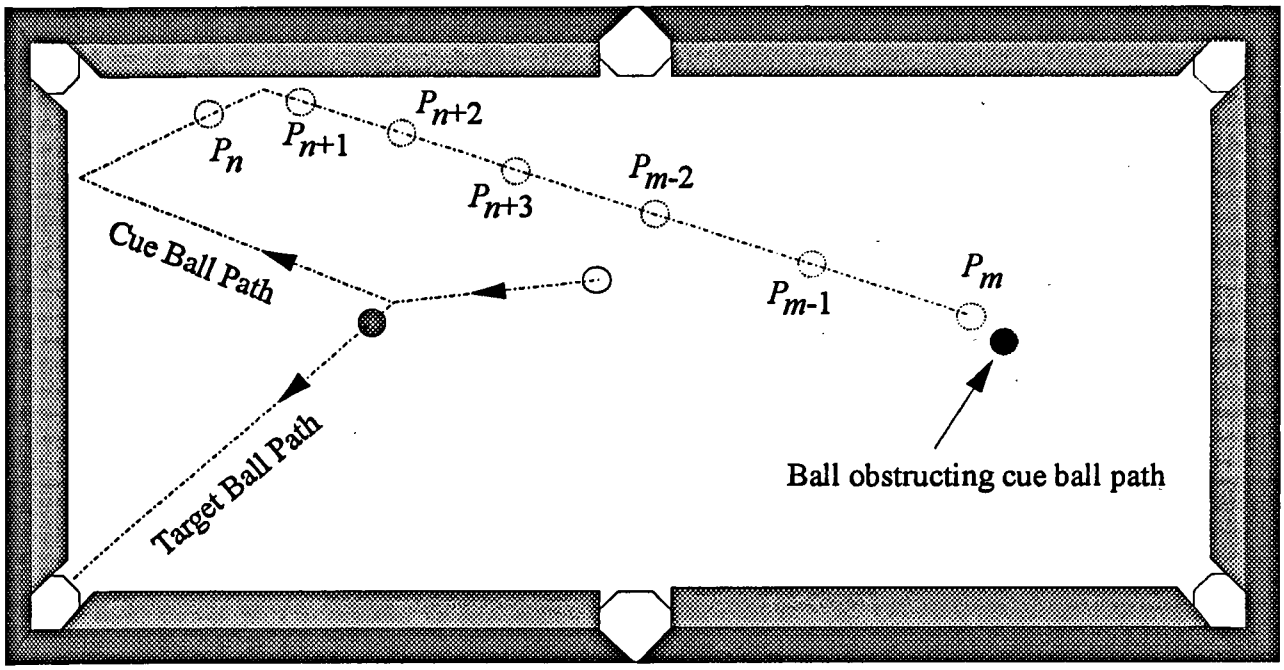
S : (*Sx*, *Sy*) Striking Ball Position
I : (*IMGx*, *IMGy*) Imaginary Ball Position
T : (*Tx*, *Ty*) Target Ball Position
P : (*Px*, *Py*) Pocket Position
$|L1| = d1$ $|L2| = d2$
→ Instantaneous Velocity
v_1 = target ball velocity immediately after impact with striking ball such that at P, the target ball velocity equals zero
v_2 = striking ball velocity immediately before impact with target ball
v_3 = initial striking ball velocity

Figure 6.1 Determination of Minimum Striking Ball Velocity for a General Shot




```
; Given :
; d = distance travelled by the snooker ball
; Vmax = initial ball velocity corresponding to pneumatic cue force 256
; Vtarget = snooker ball velocity after travelling distance d
; FIND
; initial ball velocity Vinit such that the ball velocity after travelling distance
; d equals Vtarget
tolerence = 0.5
lower = 0
upper = Vmax
REPEAT
    Vinit = (lower + upper)/2
    found = f (Vinit, d, Vfinal)
    IF Vfinal < Vtarget THEN lower = Vinit
                        ELSE upper = Vinit
UNTIL found AND Vfinal > (Vtarget-tolerence)
        AND Vfinal < (Vtarget + tolerence)
```

Figure 6.2 Pseudo Code for Ball Velocity Binary Search



Minimum cue force to pot ball = n Maximum cue force to pot ball = m

Usable cue force = i where $n <= i <= m$; $n,m,i \in$ interger ; $n <= m <= 256$

P_i = final cue ball position when played with cue force i

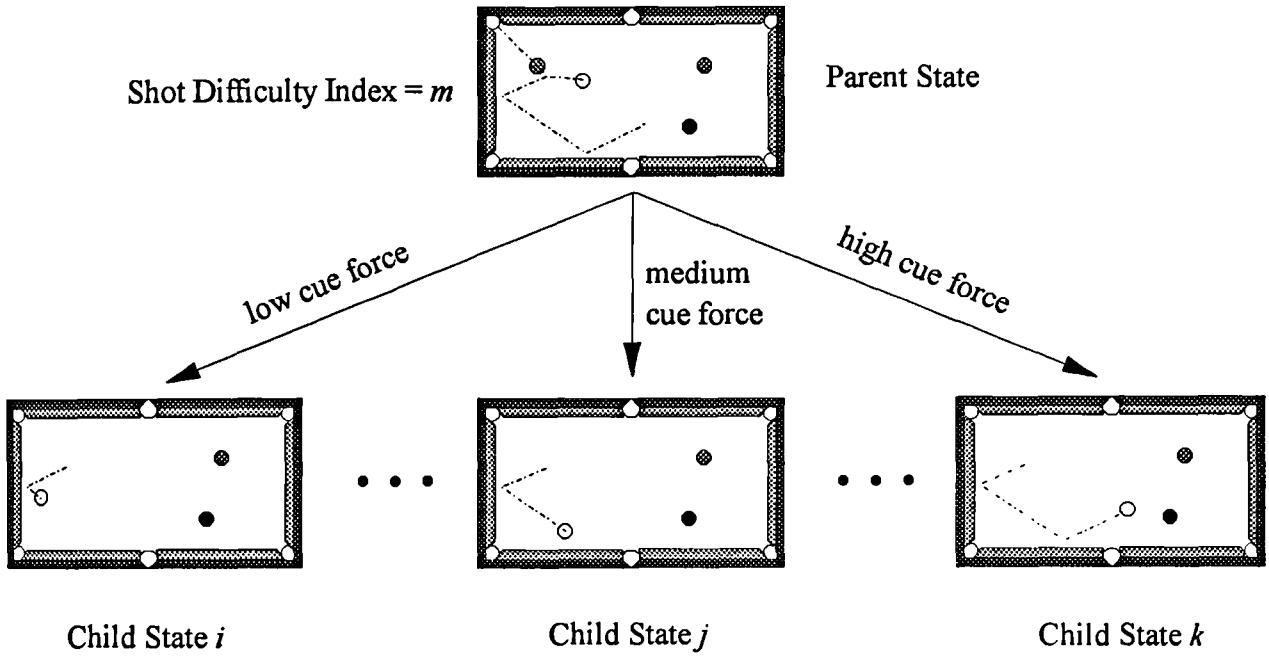
Note : $m = 256$ if no obstruction to cue ball path exists

Format of Cue Force / Cue Ball Position Map for a general shot

Cue Force	Cue Ball Position (x, y) in WORLD co-ordinates
1	null
2	null
3	null
.	.
.	.
n	P_n
n+1	P_{n+1}
n+2	P_{n+2}
.	.
.	.
m-2	p_{m-2}
m-1	p_{m-1}
m	p_m
.	.
.	.
254	null
255	null
256	null

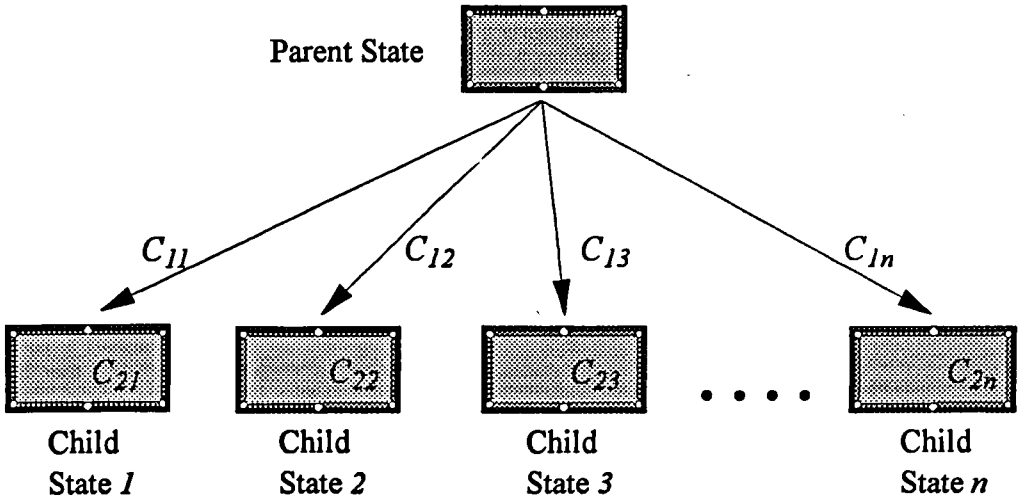
A 'null' entry means that the shot cannot be taken at the cue force corresponding to that entry.

Figure 6.3 Sample Mapping Between Cue Force and Cue Ball Position After a Shot



Note :
1. At the parent state, cue ball path is predicted using the ball dynamics model.
2. Number of child state equals number of cue forces at which the ball can be potted.
3. In the above example, the cost to transform from the parent state to a child state equals *m*.

Figure 6.4 Generation of Child States from a Parent State



Note :

1. n = total number of possible (target ball, pocket, cue force) combinations at the parent state.
2. Each (target ball, pocket, cue force) combination results in a unique child state, which is generated using the ball dynamics model.
3. C_{1i} = difficulty index of the i^{th} (target ball, pocket, cue force) shot.
4. C_{2i} = difficulty index of the best shot at child state i.

Figure 6.5 Sample Snooker Game Tree

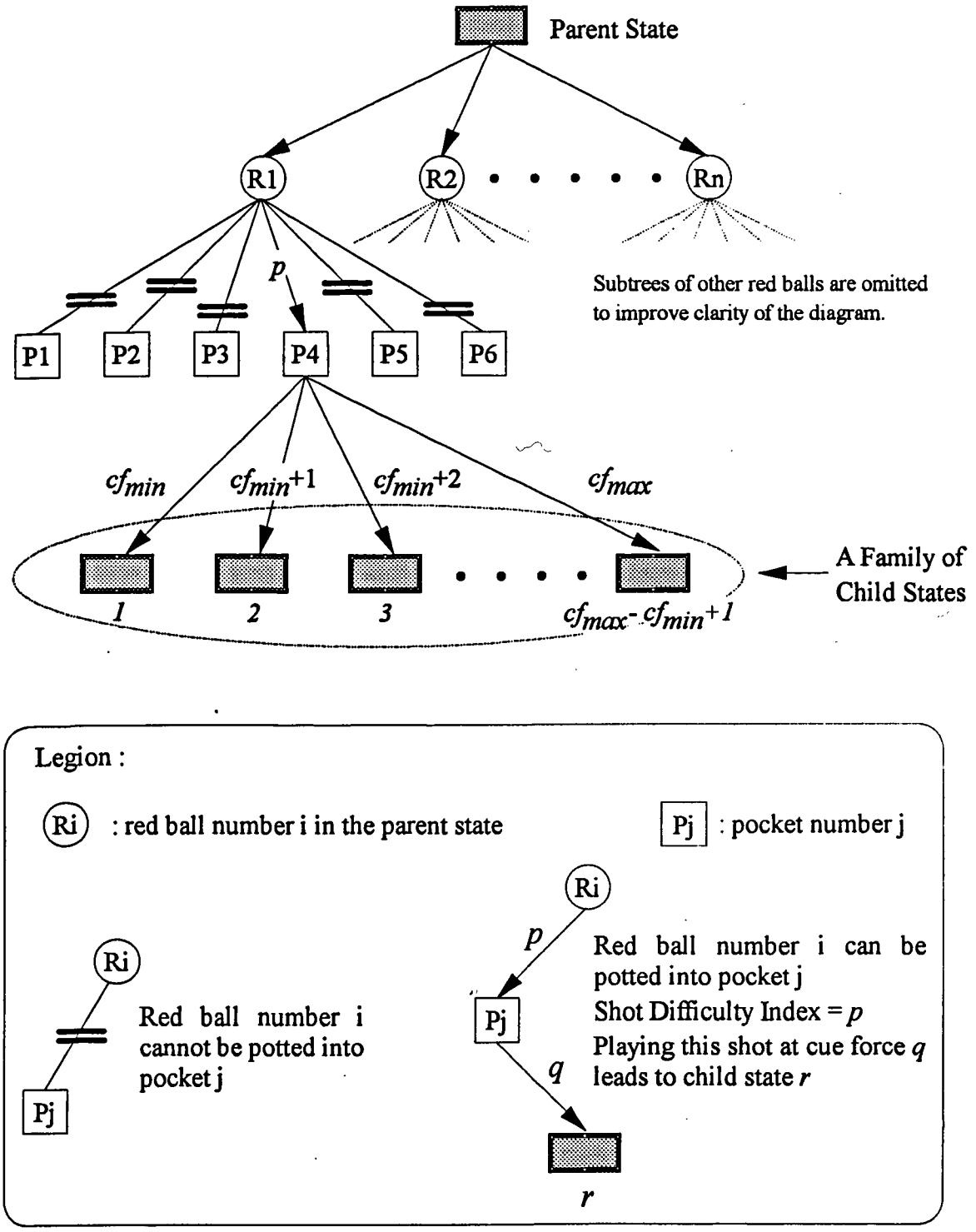


Figure 6.6 Exhaustive Search of All Red Ball/Pocket Permutations Within the Parent State and the Generation of Child States

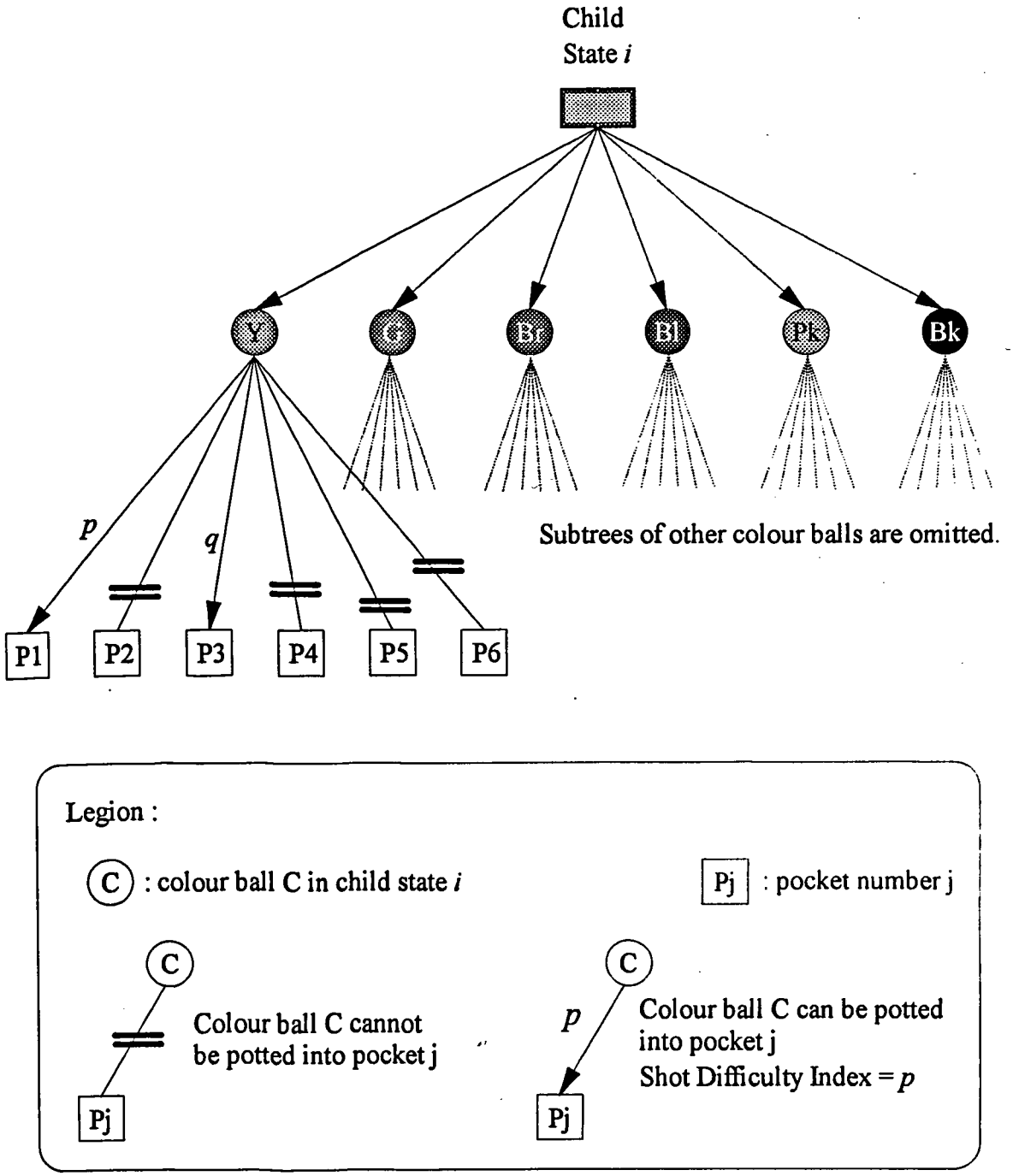
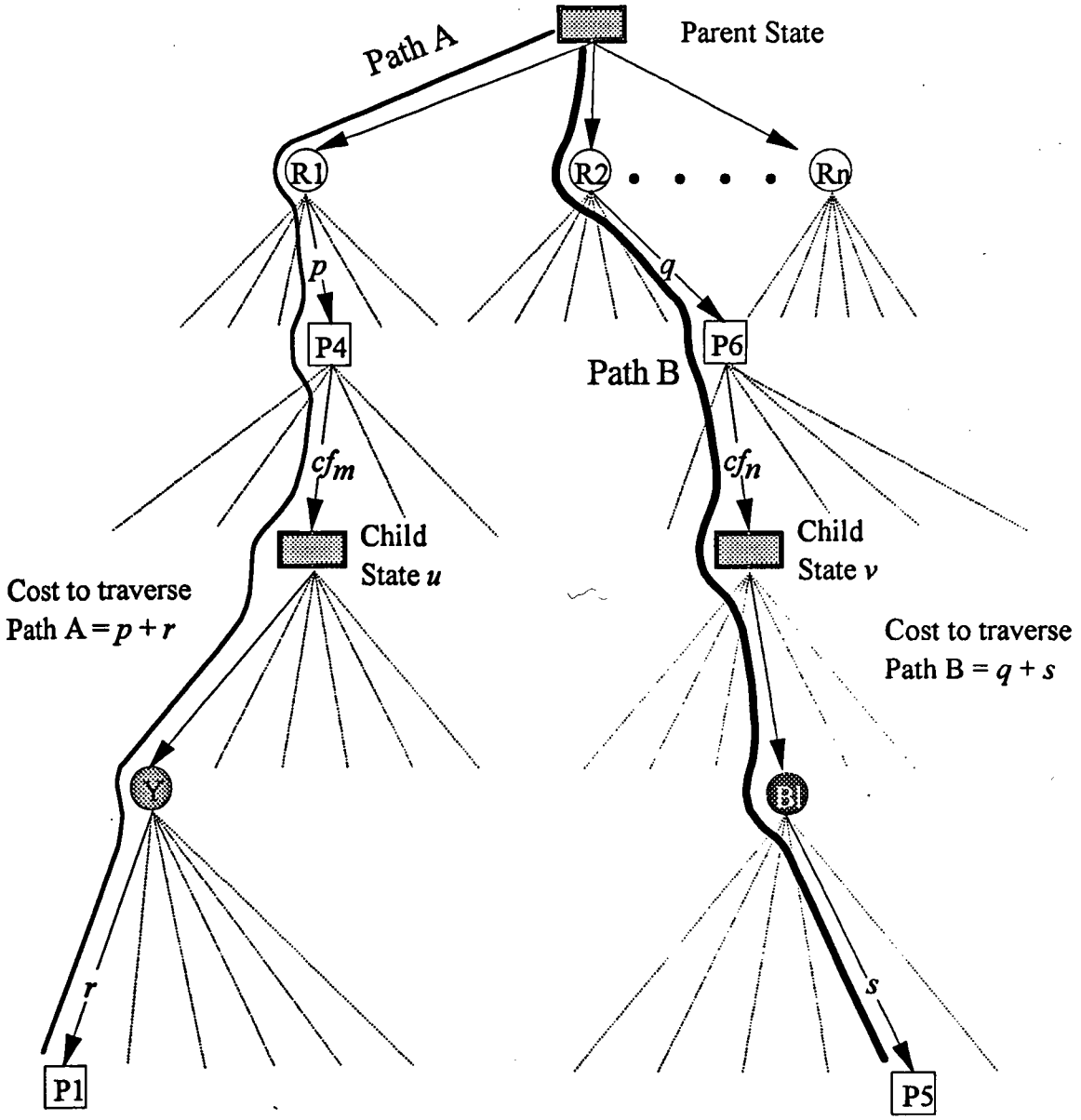


Figure 6.7 Exhaustive Search of All Colour Ball/Pocket Permutations



At the parent state :
p = Shot Difficulty Index of potting red ball number 1 into pocket 4
Playing this shot at cue force cf_m results in child state u

q = Shot Difficulty Index of potting red ball number 2 into pocket 6
Playing this shot at cue force cf_n results in child state v

At child state u :
r = Shot Difficulty Index of potting yellow ball into pocket 1

At child state v :
s = Shot Difficulty Index of potting blue ball into pocket 5

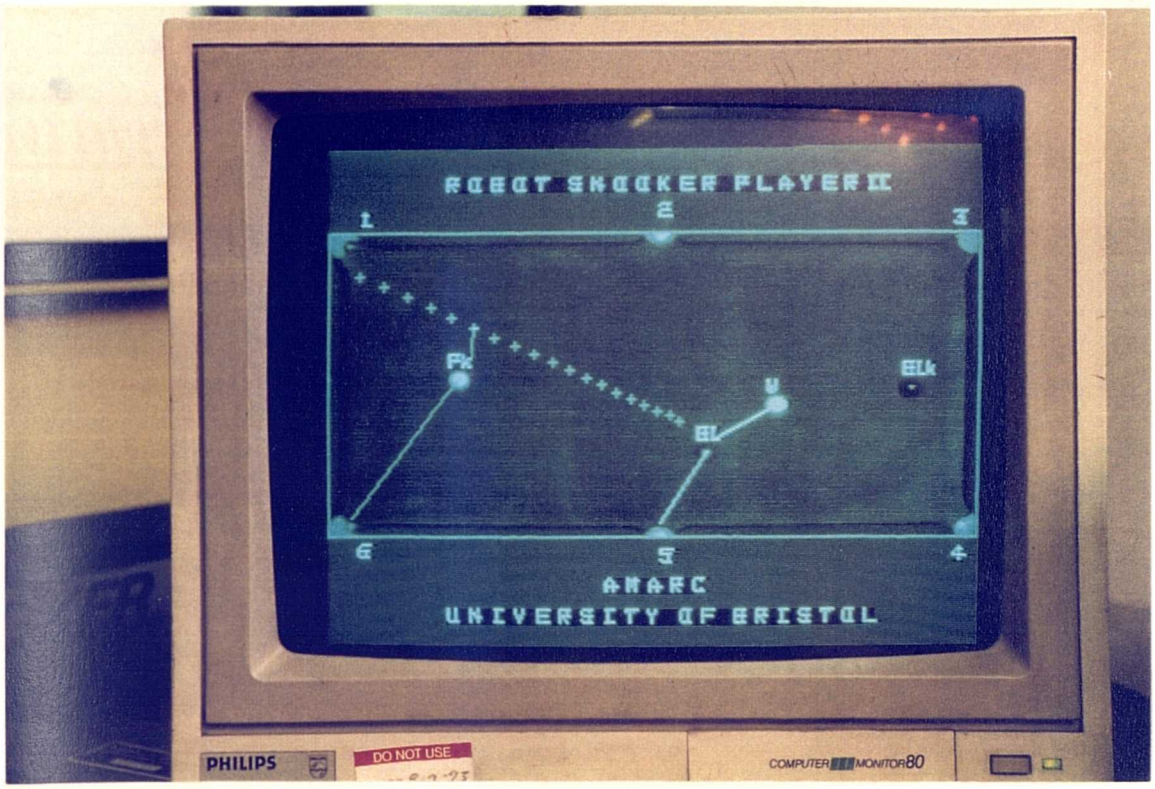
Figure 6.8 Traversal of a Sample Game Tree

ARRAY shotseq [10,2]

	1	2	
1	BOOLEAN	BOOLEAN	Flag : TRUE = shot found
2	BOOLEAN	BOOLEAN	Flag : TRUE = target ball is red
3	INTEGER	INTEGER	Ball Identification Number
4	REAL	REAL	Target Ball X Co-ordinate
5	REAL	REAL	Target Ball Y Co-ordinate
6	INTEGER	INTEGER	Pocket Identification Number
7	REAL	REAL	Shot Difficulty Index
8	REAL	REAL	Cue Ball X Co-ordinate
9	REAL	REAL	Cue Ball Y Co-ordinate
10	INTEGER	'NOT USED'	Cue Force (1-256)

Note :
The first column of the array specifies the current shot to played, after which the shot specified in the second column may be played.

Figure 6.9 Shot Plan Data Structure



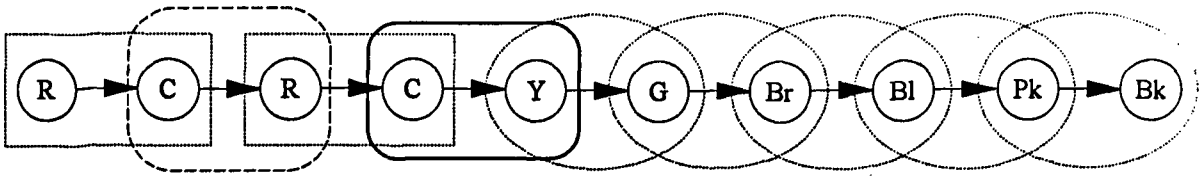
Notes :

(1) First Shot : Blue ball into pocket no. 5

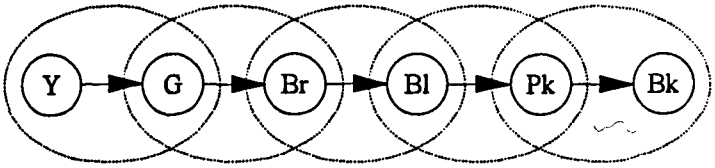
(2) Second Shot : Pink ball into pocket no. 6

(3) Each cross in the photograph is a possible finishing position of the cue ball after potting the Blue ball into pocket no. 5.

Figure 6.10 Sample Shot Plan



Sample potting sequence of a total clearance where there are 2 red balls and all the colour balls on the snooker table.
Note that any colour ball potted following a red is replaced on the table.



Sample potting sequence of a total clearance where there are no red ball and all the colour balls on the snooker table.
The colour balls have to be potted in this order.

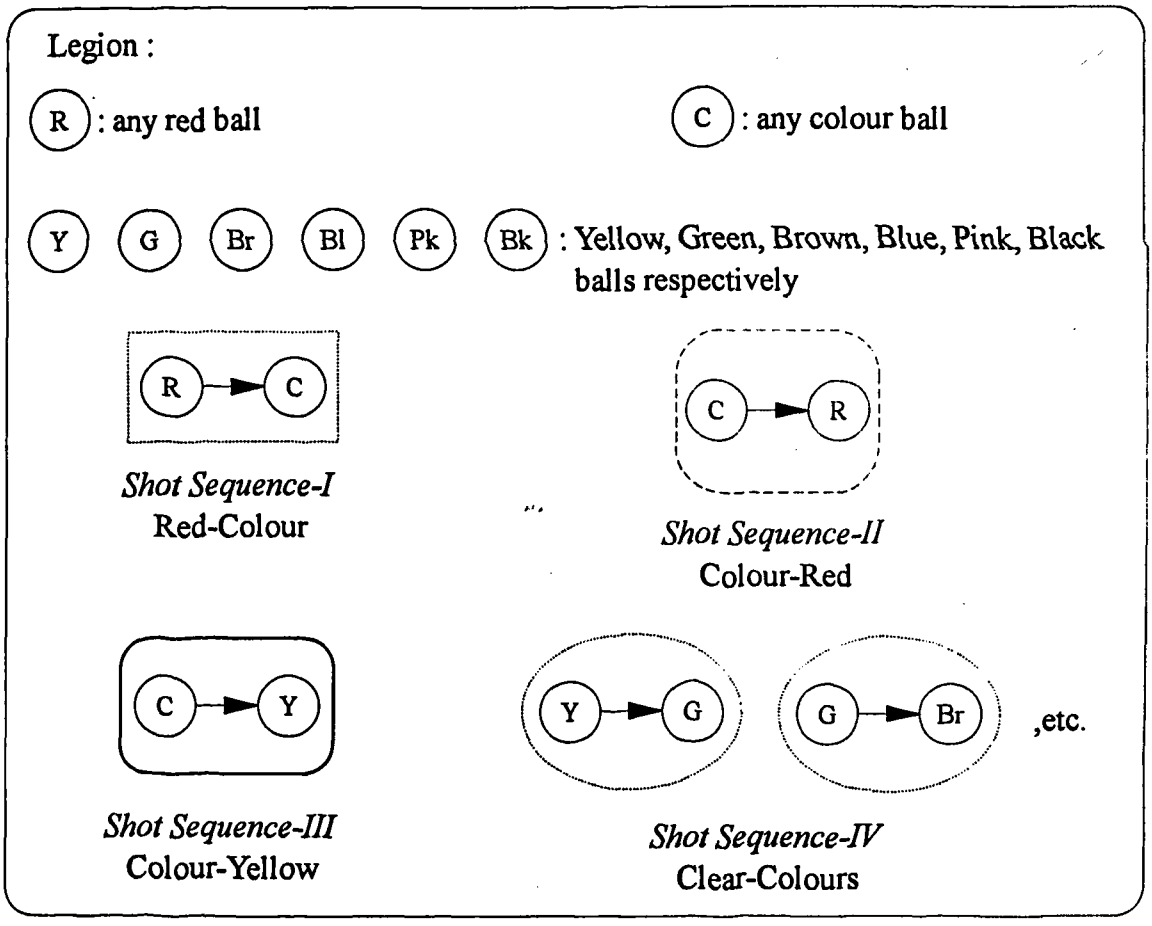


Figure 6.11 Identification of All Potting Sequences

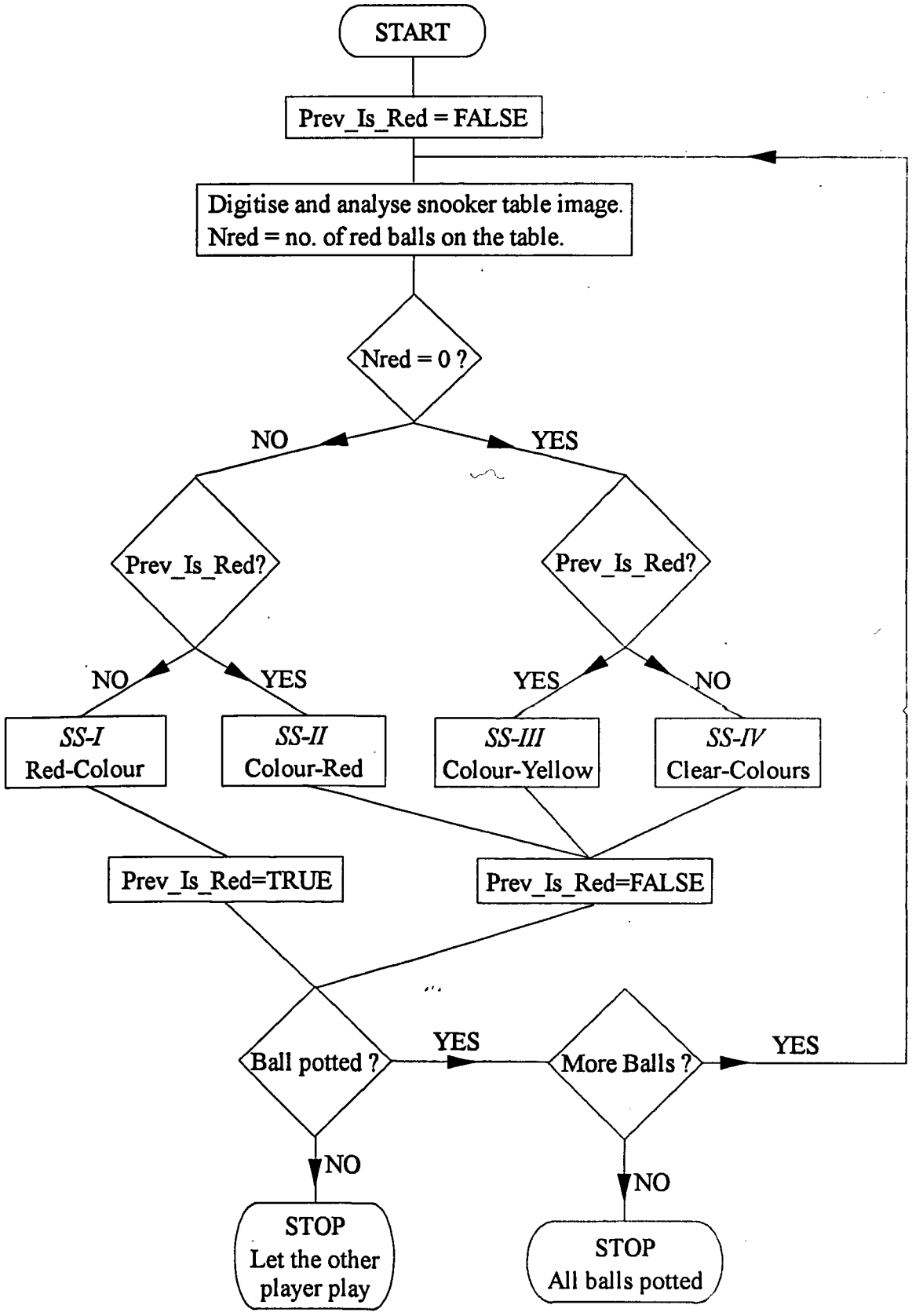
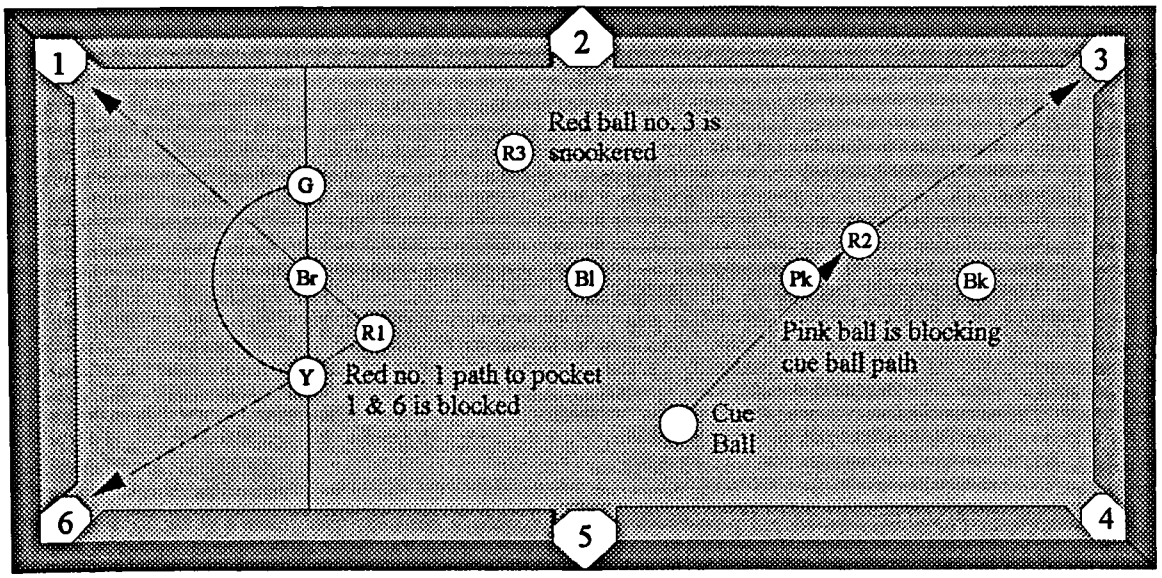
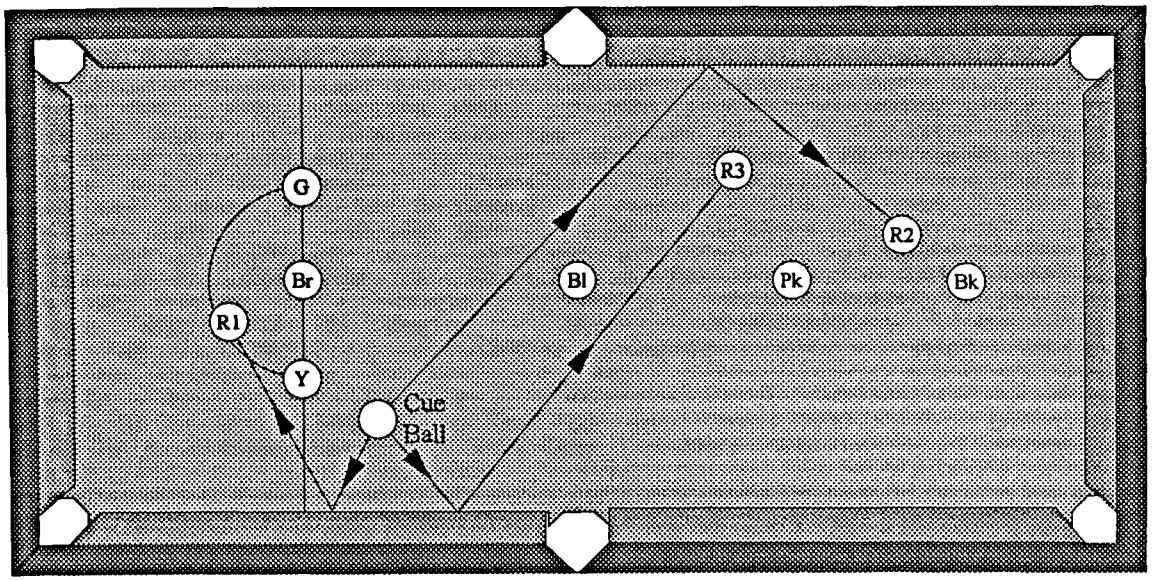


Figure 6.12 Flow Diagram of Shot Sequence Determination



No red ball can be potted but red no. 1 & 2 can be hit directly.

(a) Non-Snookered No-Pot Situation



All red balls are snookered and cannot be hit directly.

(b) Snookered No-Pot Situation

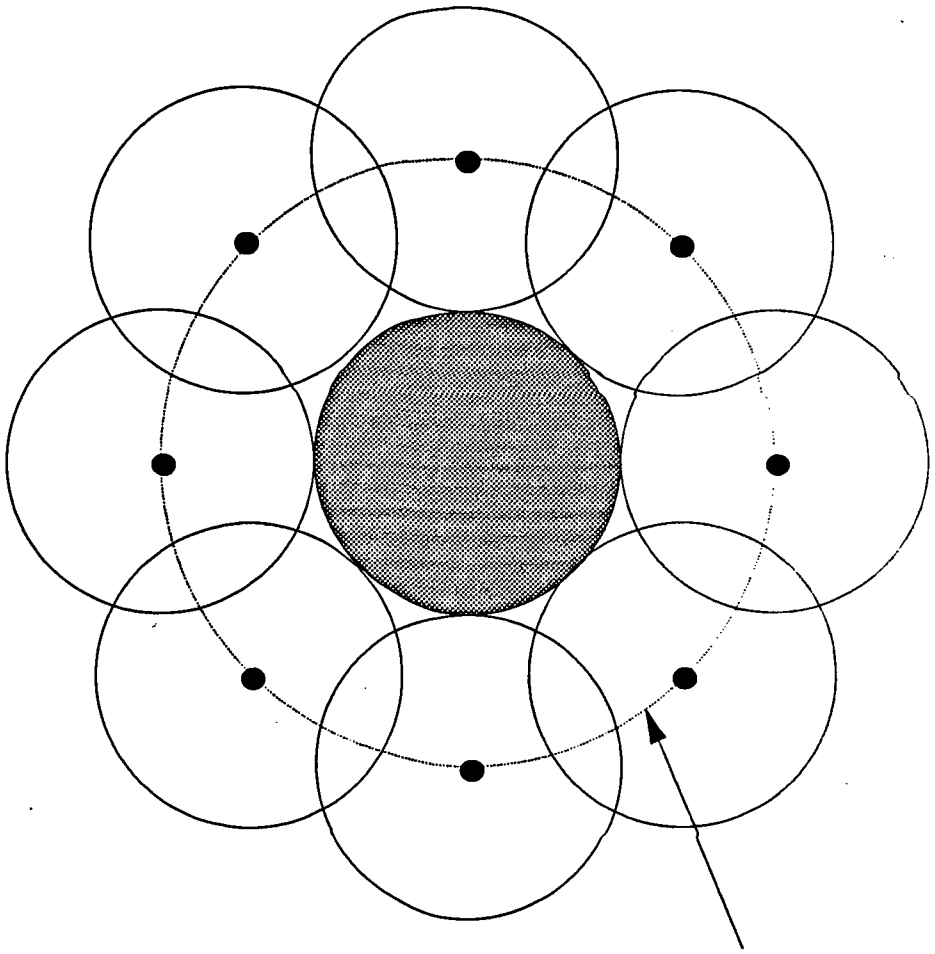
Figure 6.13 Samples of No-Pot Game Situations

N.B. : In both examples, the sequence of play is Red-Colour.

ARRAY safety_shot [7]

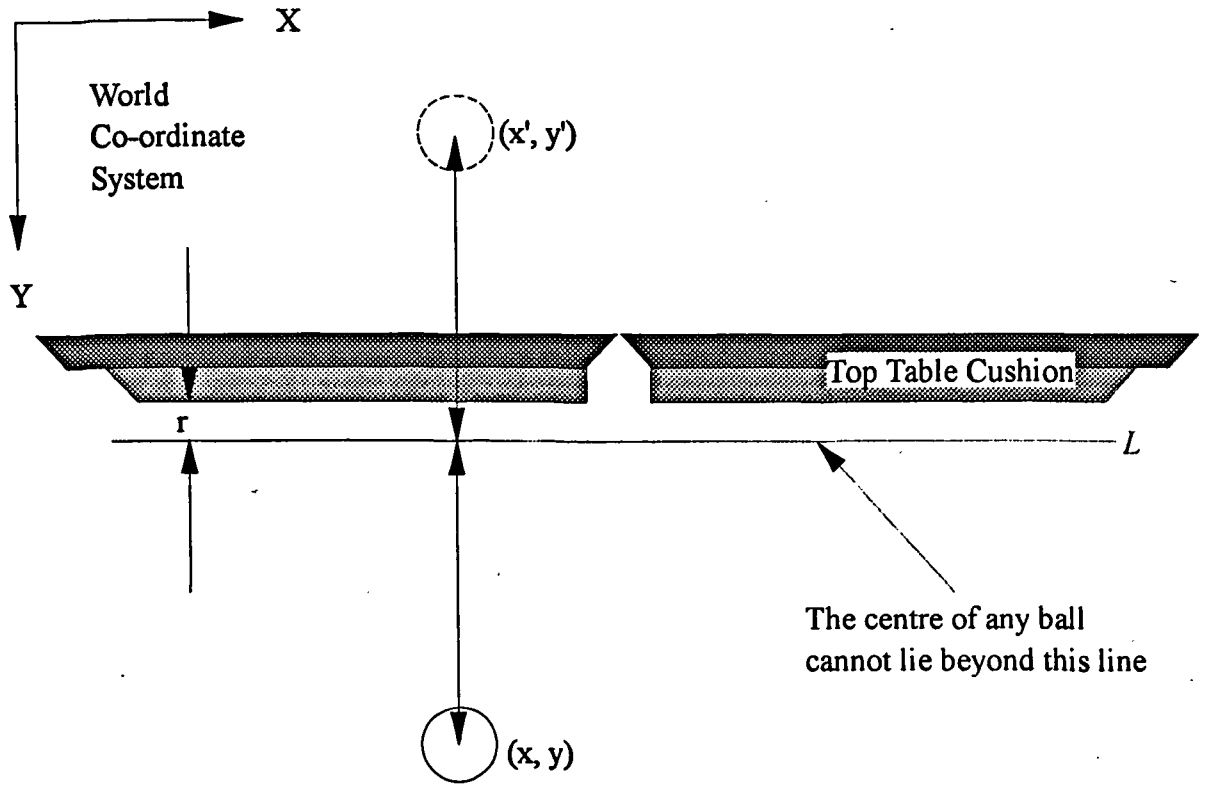
1	BOOLEAN	*Flag : TRUE = safety shot found*
2	BOOLEAN	Flag : TRUE = target ball is red
3	INTEGER	Ball Identification Number
4	INTEGER	Cushion Number
5	REAL	X co-ord. of cue ball at impact with cushion
6	REAL	Y co-ord. of cue ball at impact with cushion
7	REAL	Length of cue ball path before hitting target bal

Figure 6.14 Safety Shot Data Structure



N.B. Only 8 instead of 90 cue ball positions at impact with the target ball are shown in order to improve the diagram clarity.

Figure 6.15 Locus of Cue Ball Centres Around a Target Ball



r = ball radius
The actual ball at (x, y) and the mirrored ball at (x', y') are equi-distance from line *L*.

Figure 6.16 Using the Cushion as a Reflective Surface

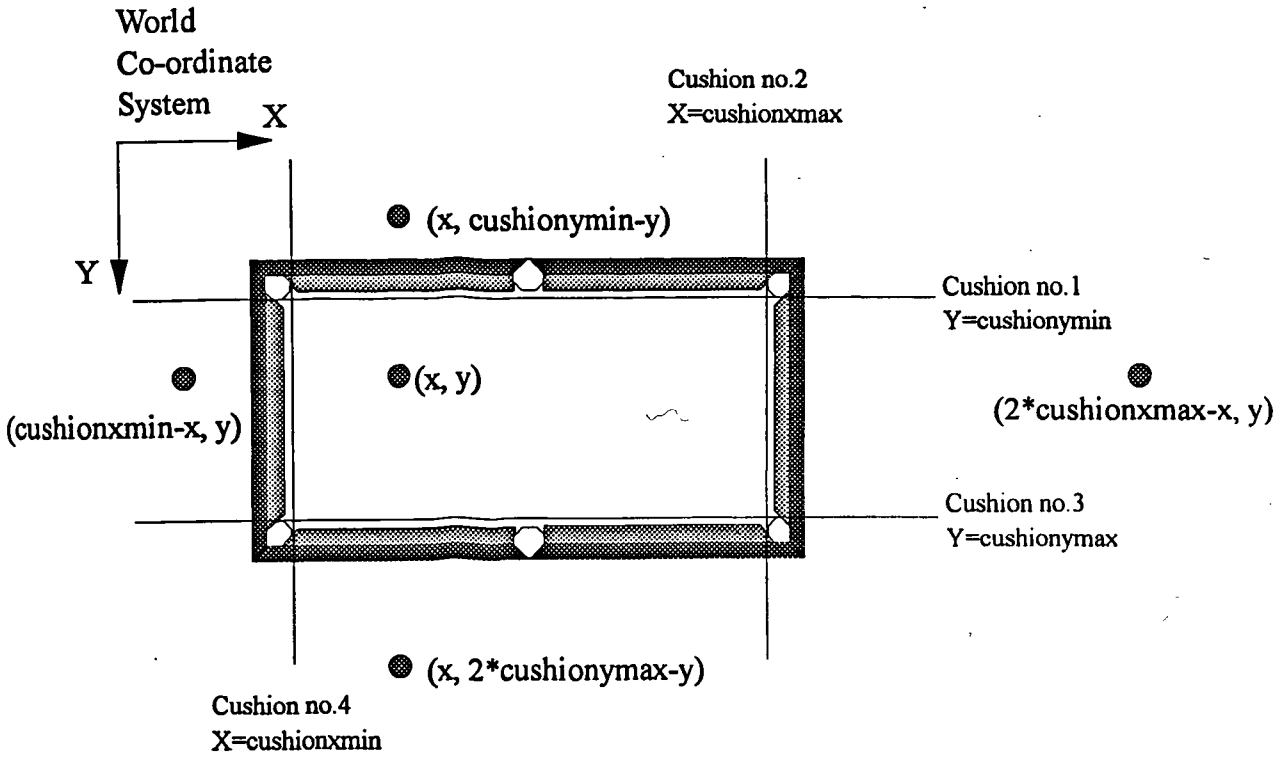


Figure 6.17 Modelling of the Cushions as Lines Parallel to the World Axes and the Determination of the 4 Mirrored Positions of a Ball at a general position (x, y)

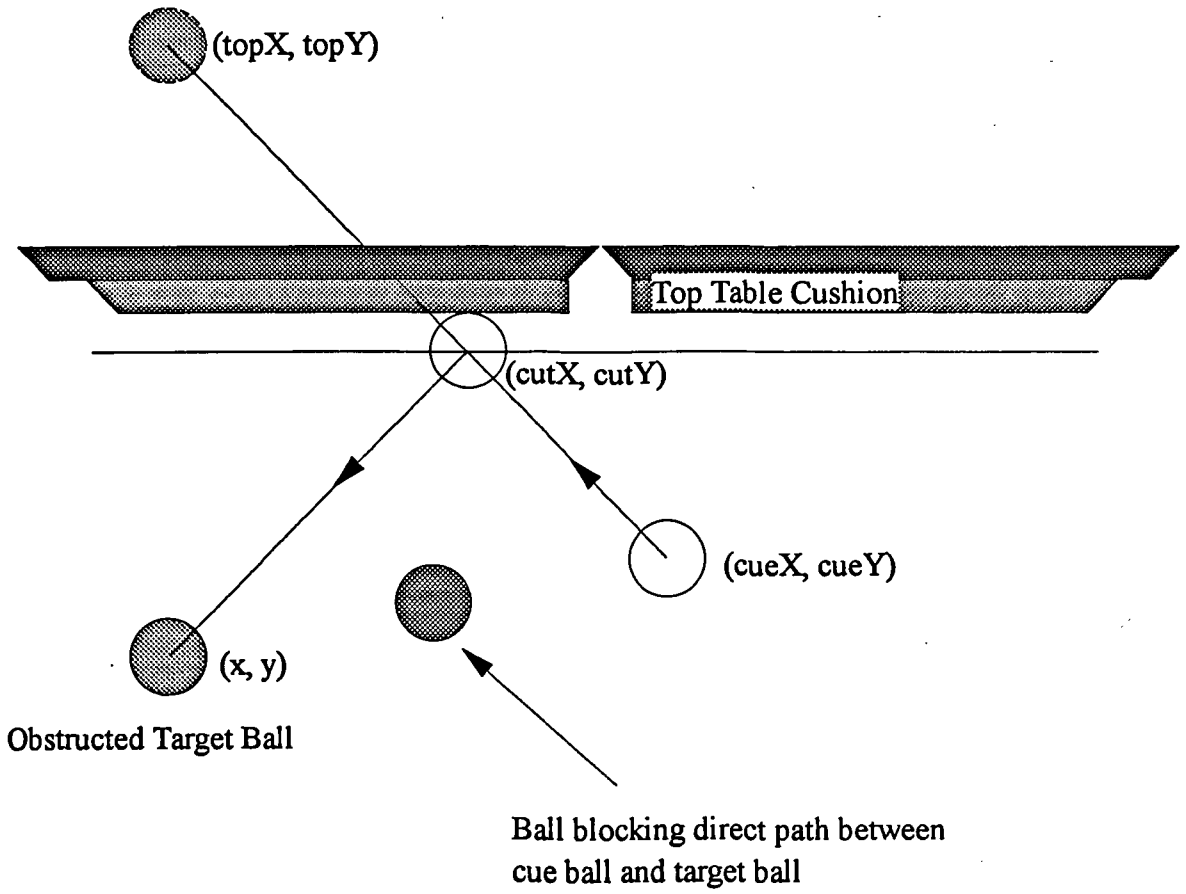
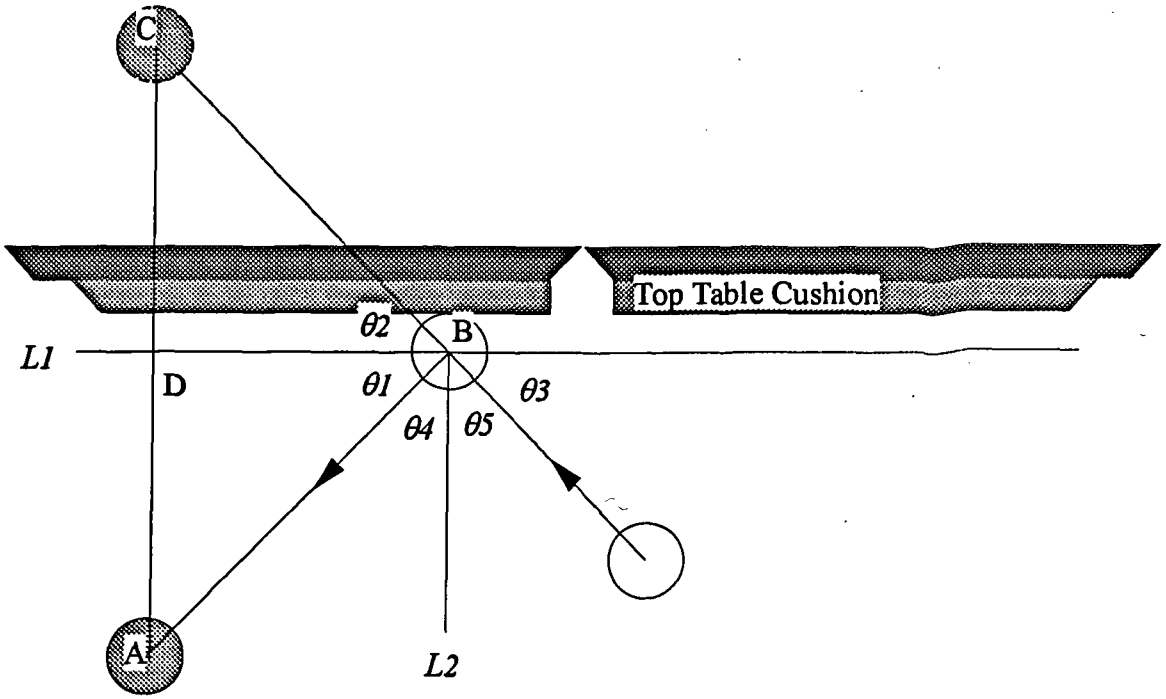


Figure 6.18 Determination of Impact Point Along The Top Cushion in order to hit the Obstructed Target Ball



Given : L1 and AC are perpendicular,
L1 and L2 are perpendicular,
and AD = CD

It can be deduced that : $\theta 1 = \theta 2 \Rightarrow \theta 3 = \theta 1$

And since $\theta 1 + \theta 4 = \theta 3 + \theta 5 = 90^\circ$

Therefore $\theta 4 = \theta 5$ as $\theta 3 = \theta 1$

$\therefore$

i.e. the reflected angle equals the incident angle

Figure 6.19 Proof that the Reflected Angle Equals the Incident Angle

Chapter 7 Optimisation of Snooker Ball Dynamics Modelling Using Genetic Algorithm

The performance of the game planning module discussed in the previous chapter is critically dependant on the accuracy of the ball dynamics model discussed in chapter 3. If the predicted ball motion were to closely match the actual ball motion, parameters such as the coefficient of friction, restitution, etc. must be accurately determined. Figure 7.1 is a diagrammatic illustration of how different sets of parameters lead to different predictions of snooker ball motion given the same shot data. The ultimate aim is to determine dynamics model parameters such that the prediction best matches the true ball motion.

Direct determination of each parameter through experimentation appears to be the most obvious answer. Indeed experiments were conducted to determine the values of the parameters. However, it was decided that a method to determine their values not involving direct measurement would be more desirable for a number of reasons. First of all, the setting up of experiments is laborious and time consuming. Some experiments e.g. that conducted to determine the coefficient of restitution of ball-to-cushion impact requires the use of high speed video equipment which has to be borrowed from the SERC months in advance. Secondly, the test results are only correct at the time the experiments were conducted. Any subsequent variations of these parameters due to normal wear and tear of the table or temperature variation, for example, will not be accounted for. Finally, the whole set of experiments will have to be repeated if the system were to be transported to a different snooker table.

Enumerative and random search methods are also ruled out because of the vast dimension of the search space. These methods are most effective in applications where the performance of the modern computer is high enough to overcome the complexity of the problem. Unfortunately, to determine the values of 5 real parameters through enumeration or random search is considered too inefficient.

Genetic algorithms are search algorithms based on the mechanics of natural selection and genetics. The origin of GA can be traced back to the late 50's when researchers such as Baricelli **[Barricelli62]** and Rosenberg **[Rosenberg67]** used computers to simulate natural genetics. In those applications, computers were used merely as tools to aid the understanding of genetics. In the early 70's, Holland **[Holland75]** started experimenting with the idea of solving complex computational problems by replicating nature's method of evolution in algorithmic form. Natural biological systems exhibits many features which Holland found desirable, they include the ability of self-repair, self-guidance, and reproduction. Background information on genetic algorithms can also be found in

[Davis90]. The mechanism of genetic algorithms will become clear as its application to the problem of ball dynamics modelling is discussed below.

7.1 Coding of Ball Motion Parameters as a Chromosome

In modelling the dynamic behaviour of snooker balls, the values of the following 5 parameters (units in brackets where applicable) have to be known :

(1) ***Cue_Ball_Vmax*** *(mm/s)* = maximum cue ball velocity corresponding to pneumatic cue force level 256

(2) ***Mu*** = coefficient of friction between snooker ball and snooker table

(3) ***Omega*** *(mm/s²)* = rolling resistance exerted by snooker table on balls expressed as the equivalent linear deceleration

(4) ***Rest_BB*** = coefficient of restitution of ball-to-ball impact

(5) ***Rest_BC*** = coefficient of restitution of ball-to-cushion impact

The above parameters are to be coded as a binary string. Experiments indicate that the ranges of the parameters are as follow :

$$2000.0 \leq Cue\_Ball\_Vmax \leq 5000.0$$

$$0.05 \leq Mu \leq 0.5$$

$$50.0 \leq Omega \leq 400.0$$

$$0.5 \leq Rest\_BB \leq 1.0$$

$$0.5 \leq Rest\_BC \leq 1.0$$

Having defined the ranges of the parameters, the number of bits allocated to represent each parameter has to be decided. An extreme example is to allocate a single bit to represent the value of ***Cue_Ball_Vmax***, such that the binary string '0' and '1' correspond to a value of 2000.0 and 5000.0 respectively. This coding is clearly inappropriate but it does show the effect of binary string length on the precision of a parameter. An exceedingly long binary string is equally inappropriate because there is no real benefit in coding the parameter to a precision in the order of 10^{-57}, for example, by using 200 bits.

As a guideline, parameters which cannot exceed 1.0 are coded to a precision of under 0.001 while those exceeding 1.0 are coded to a precision of under 0.01. The binary codings of all parameters are summarised in table 7.1. The table lists the 5 parameters in the order they are concatenated to form a chromosome of length 45 bits. Thus the first 15 bits in a chromosome correspond to the parameter *Cue_Ball_Vmax*, the next 6 bits correspond to the coefficient of friction *Mu*, etc. All parameters are coded as unsigned real numbers.

Since the demarcations of the constituent binary string representations of the parameters and their respective lower and upper limits are known, the values of the parameters coded into any general chromosome can be retrieved. This is best illustrated through a worked example to extract the value of the coefficient of friction *Mu* encoded in a sample chromosome. Let the chromosome be the string below :

string position 16
↓
'101000101110101010111010101010001001101000111'

Table 7.1 shows that the binary representation of *Mu* starts at string position 16, is 6 bits long and the minimum and maximum values of *Mu* are 0.05 and 0.5 respectively. This allows the substring corresponding to *Mu* to be extracted from the chromosome as :

'010111' = 23 decimal

The largest 6-bit binary number is :

'111111' = 63 decimal

Thus the value of *Mu* as coded in the chromosome can be retrieved as :

$$Mu = (\max - \min)\left(\frac{decimal('010111')}{decimal('111111')}\right) + \min$$

$$= (0.5 - 0.05)\left(\frac{23}{63}\right) + 0.05$$

$$= 0.214$$

given that the function '*decimal*' converts an unsigned binary number into its decimal equivalent and that the min and max values of *Mu* are 0.05 and 0.5.

The values of the other parameters are similarly retrieved from a chromosome. Note that the reverse conversion is not necessary because genetic algorithms work upon chromosomes, i.e. binary codings of the parameters, rather than changing the values of the parameters directly as in random search algorithms. In this sense, genetic algorithms are 'blind' to the problems they are trying to solve.

At this point, there is no need to be concerned with the actual values represented by the binary codings. The important thing is to realise that the 5 binary strings corresponding to the 5 ball motion parameters can be appended together to form a single binary string of length 45 bits, which can be referred to as a *chromosome*. It follows that there can be a maximum of 2^{45} distinct chromosome, each corresponding to a unique solution set. Each 'digital' chromosome can be subjected to a variety of genetic operators, just like its biological counterpart. Suppose there exists an optimal solution set to the dynamic modelling problem, the corresponding binary coding must be a member of the set of all possible chromosome permutations, allowing for parameter quantisation errors.

7.2 Generation of an Initial Population

Before generating a population, the population size has to be decided first. In general, a moderate population size of at least 30 individuals is advisable. It is therefore decided that a population of 40 individuals will be maintained in each generation. The term individual will be used interchangeably with the term chromosome, both referring to a binary string of 45 bits.

Genetic algorithms falls into the category of directed random search algorithms and this fact is reflected in the random generation of the initial population. The initial value of each bit of every individual is either a '0' or a '1' which is randomly selected according the value of a pseudo-random number *RND*, $0 \leq RND \leq 1$. To achieve a 50:50 chance of getting a '0' and a '1', a bit is assigned '0' if *RND* is less than 0.5, '1' otherwise. This process is equivalent to tossing an unbiased coin. A random individual can therefore be created after 'tossing the coin' 45 times. This process is in turn carried out 40 times to create a random initial population containing 40 individuals. Table 7.2 is a sample random initial population. The random initial population is a significant feature of genetic algorithms. In this application, 40 random initial points in the search space are created which form the starting points of a parallel search for the optimal.

7.3 Assessing the Fitness of Chromosomes

Having created a random initial population, a means of assessing the fitness of each individual within the population is required so that individuals can be differentiated

according to their fitness. Unlike some problems in which candidate solutions can be fed directly into an objective function for assessment, testing the correctness of a set of motion parameters demand a more complex method.

Given a chromosomal representation of a solution set, the real values of the 5 motion parameters can be retrieved. As the optimal parameter values are unknown (otherwise there will be no need for the work discussed here), the correctness of the decoded parameters cannot be directly compared with the optimal and thus assessed. However, assuming that the ball dynamics model and the ball motion parameters are correct, the predicted path of the cue ball after a shot (see chapter 6) will match the actual cue ball path when the shot is taken by the robot. Consequently, the discrepancy between the actual and predicted cue ball path is a measure of the correctness of the parameters. The degree of correctness is inversely proportional to the magnitude of the discrepancy.

Before a shot is taken by the robot, the position of the cue ball at impact with the target ball, P_{start} is pre-determined as part of the shot data. The final position, P_{stop}, of the cue ball after the shot is taken can be determined using the AUTOMATIX AV5 vision system. The intermediate points, P_{int}'s, along the cue ball path, however, cannot be determined by the AV5 as it is incapable of tracking the motion of a snooker ball in real time. To record these intermediate points, talcum powder (which is very fine and so does not affect ball motion in any way) is sprinkled on parts of the table where the cue ball is likely to hit the cushion after hitting the target ball. As the cue ball travels past the area covered in talc powder, the contact points with the cushions will show up clearly. A snooker ball can then be positioned at each of these marked points, after which the snooker table image is digitised and their positions in world co-ordinates computed. The actual cue ball path defining points, i.e. P_{int}'s, and P_{stop}, together specify the complete path of the cue ball after impact with the target ball as a collection of line segments. For any general shot, P_{start} is invariably the starting point of the cue ball path after impact and therefore need not be stored as a path defining point. Figure 7.2 shows an example cue ball path.

Using the above method, the dynamic behaviour of the cue ball in a shot can be recorded given a set of shot data. The shot data includes the pocket position, the target ball position, the cue ball position and the pneumatic cue force used to take the shot. The *Actual Path Data* includes a 1-dimensional array *Actual_Path_List* for storing the actual cue ball path defining points and a counter *Actual_Count* of the number of path defining points. The dimension of the array is set to 12 (sufficient to store 6 (x, y) co-ordinates) as the cue ball is not expected to hit the cushions more than 5 times before coming to rest due to limitation in the maximum pneumatic cue force. Referring back to figure 7.2, the *Actual Path Data* are :

$$Actual\_Path\_List = [P4_x, P4_y, P5_x, P5_y, P6_x, P6_y, 0, 0, 0, 0, 0, 0]$$

$$Actual\_Count = 3$$

Being able to record the actual path of the cue ball on taking a shot allows the correctness of the output of the ball dynamics model to be assessed. This is achieved by feeding the same set of shot data into the model together with a set of ball motion parameters. The detail generation of the predicted cue ball path is already covered in chapter 6. The main concern here is that the output of the model is a set of *Predicted Path Data* which is in the same format as the *Actual Path Data*, comprising of an array *Predicted_Path_List* and a counter *Predicted_Count*.

The next task is to quantify the difference between the actual and the predicted path of the cue ball given the same set of shot data. For any general shot, the actual path of the cue ball is measured by physically taking the shot using the robot. Hence the actual path is a record of what actually happened in reality. On the other hand, the predicted path is generated by the ball dynamics model whose performance is dependant on the accuracy of the various motion parameters.

Performance assessment is most straightforward if *Actual_Count* equals *Predicted_Count* i.e. the model successfully predicted the correct number of contacts between the cue ball and the cushion, as in figure 7.3. Performance of the prediction generated by the model is measured as the sum of the deviations between the corresponding points in *Actual_Path_List* and *Predicted_Path_List*.

Because of parameters variations from their correct values, it is possible that the predicted path is so different from the actual path that the counter values are different. The equation below computes the performance figure of a predicted path irrespective of whether the counters are equal or not :

Given : $APL = Actual\ Path\ List$ $AC = Actual\ Count$

$PPL = Predicted\ Path\ List$ $PC = Predicted\ Count$

$AC > 0$ $PC > 0$

$$\text{Performance} = \sum_{i=1}^{\min(AC,PC)} \text{distance}(APL[f1(i)], APL[f2(i)], PPL[f1(i)], PPL[f2(i)])$$

$$+ \sum_{i=\min(AC,PC)}^{\max(AC,PC)-1} \text{distance}(K[f1(i)], K[f2(i)], K[f1(i+1)], K[f2(i+1)])$$

Eq. *(7.1)*

Notes on Eq. *(7.1)* :

1. $f1(i) = 2i\text{-}1, f2(i) = 2i$
2. $f1$ and $f2$ are used to compute the array indexes of the i^{th} (x, y) co-ordinates in a path list array.
3. K is a temporary array of size 12.
4. $K = APL$ if $AC > PC$, else $K = PPL$.
5. The second summation series is not initiated if $AC = PC$ since the upper limit will be less than the lower limit.

In reality, this perfect model proves elusive for a number of reasons. First of all, the performance of the pneumatic cue is not absolutely repeatable since it is no more than a modified industrial pneumatic cylinder, which is susceptible to variations in temperature and frictional loss. Secondly, the cueing accuracy is dictated by the accuracy of the PUMA robot arm and the SKF linear driver. The model assumes that the cue ball is always hit horizontally through the centre. However, hitting the cue ball 2mm, for example, above and below the centre line produces significant variation in the behaviour of the cue ball, particularly at high cue forces. Thirdly, the diameter of the snooker balls varies by up to 0.5mm thus their weights, assumed to be equal in the model, also vary. Finally, the properties of the snooker table i.e. *Mu* and *Omega*, assumed to be constant, are affected by temperature and humidity changes. Also, the table cloth and cushions, assumed to be homogeneous, are not necessarily so but to model the table as anything other than that would be impractical.

Since the values of 5 unknown parameters have to be determined, at least 5 different shots have to be taken to ensure that a unique solution will be found. Also, in at least 1 of the example shots, the cue ball must hit the table cushion at least once to ensure that the coefficient of restitution of ball-to-cushion impact plays a part in the cue ball motion. The 5 example shots are presented in Appendix D. To assess the fitness of a chromosome, it is first decoded resulting in the real values of the 5 ball motion parameters. For each of the 5 example shots, the corresponding shot data and the set of ball motion parameters are fed into the ball dynamics model, producing a predicted cue

ball path which is then compared with the actual path as described above. The fitness of the chromosome is thus measured as the total deviation. This process of fitness assessment with reference to the 5 example shots is summed up in figure 7.4.

7.4 Parents Selection

One of the fundamental rules of genetic algorithms is that the probability of any individual within a population being selected as a parent for reproduction is proportional to its fitness. In this application, fitness is inversely proportional to F_i which is a measure of error for the i^{th} individual in the population. As fitness is inversely proportional to F_i, its multiplicative inverse $F_i' = 1/F_i$, $F_i \neq 0$ is computed. F_i' is directly proportional to the probability of selection for the i^{th} individual.

A most effective parent selection strategy is the Biased Roulette Wheel Parent Selection. In an unbiased roulette wheel there are N numbered compartments of equal size around the wheel. The idea is to spin the wheel and then throw a ball onto it. When the wheel stops, the ball will land inside one of the compartments. There is equal probability for the ball to land in any compartment. By associating a compartment with each individual in the population such that the size of compartment i equals the individual's fitness F_i', a biased roulette wheel is created as shown in figure 7.5. High performance individuals have correspondingly large compartments on the biased roulette wheel resulting in higher chances of selection, which is in accordance with the theory of survival of the fittest and natural selection. The throwing of the ball is simulated by the generation of a random number RND normalised as RND' which lies between $[0, \sum_{i=1}^{40} F_i']$ inclusively. In general the k^{th} individual is selected for reproduction if RND' falls inside the k^{th} compartment of the biased roulette wheel, when the following condition holds :

$$\text{SumFitness}_{k-1} < RND' \leq \text{SumFitness}_k$$

$$\text{where SumFitness}_n = 0, \qquad n = 0$$

$$= \sum_{i=1}^{n} F_i', \quad 1 \leq n \leq 40$$

Using the biased roulette wheel, 40 parents are selected from the population for reproduction.

7.5 Reproduction

In genetic algorithms, reproduction is the means by which the search for the optimal solution is guided. According to Goldberg's schemata theory, performance of a chromosome can be mapped to certain substrings within it. Reproduction facilitates the passing of these high performance substrings onto subsequent generations while phasing out the poor ones, thereby improving the overall performance of the population through the GA cycle. Details of the mathematical proof of the effectiveness of GA can be found in Goldberg **[Goldberg89]**.

The 40 selected parents are sequentially paired off as couples ready for mating which involves crossover and mutation. The probability of crossover and mutation taking place is dictated by the system variables GA_CROSS_RATE and GA_MUTATE_RATE. Both are real numbers between 0 and 1. Typical crossover and mutation rates are 0.6 and 0.001 respectively, which will be adhered to in this application. Other genetic operators such as inversion and multi-point crossover exist but only crossover and mutation are implemented to keep the system pure and simple yet no less effective.

7.5.1 Crossover

Crossover facilitates the exchange of genetic material between two parents resulting in offspring carrying new combinations of genetic contents. Let the 2 parent chromosomes involved in reproduction be $Parent_1$ and $Parent_2$, both being a binary string 45 bits long. Whether crossover is to take place depends on the value of a pseudo-random number *RND*. If *RND* is greater than GA_CROSS_RATE, no crossover will occur and the offspring $Child_1$ and $Child_2$ equals $Parent_1$ and $Parent_2$ respectively. Otherwise crossover will take place. Another pseudo-random number is transformed into an integer *CrossPoint* such that *CrossPoint* $\in$ {1, 2, 3,...44}. *CrossPoint* cannot equal 45 as no chromosomal information interchange would result given that the chromosome length equals 45. $Parent_1$ is split into two strings : $Parent_{11}$ and $Parent_{12}$. $Parent_{11}$ equals *substring* ($Parent_1$, 1, *CrossPoint*) and $Parent_{12}$ equals *substring*($Parent_1$, *CrossPoint* + 1, 45-*CrossPoint*). $Parent_2$ is similarly split into $Parent_{21}$ and $Parent_{22}$. The function ' *substring* ' takes an input string, the start position and the number of bits to extract from the input string as operands and returns the specified substring. Another useful string operator is the ' *concat* ' function which concatenates two input strings as a single output string. After crossover, the offspring are constructed as :

$$Child_1 = concat\ (Parent_{11}, Parent_{22})$$

$$Child_2 = concat\ (Parent_{21}, Parent_{12})$$

7.5.2 Mutation

Mutation is the random alteration of the genetic contents of an individual. Despite its low rate of occurrence (0.001 compared to 0.6 for crossover), mutation is a very important genetic operator that introduces variety to the population and prevents the permanent loss of genetic information. Suppose the first bit of all individuals in the random initial population is a ' 0 '. Without mutation, the first bit of any individual in all subsequent populations remains a ' 0 ', thereby possibly ruling out any prospect of finding the optimal solution. The presence of the mutation operator prevents a certain site on all individuals from being stuck with the same genetic information.

Mutation operates on a single bit of a chromosome by toggling its value from ' 0 ' to ' 1 ' and vice versa. Following crossover of 2 parents, 2 children are produced. Each of the 2 children is then subjected to a bit-wise scan. On each scan, if *RND* is less than GA_MUTATE_RATE, the bit is toggled, otherwise it remains unaltered. At the selected rate of mutation, on average 1 bit out of every thousand bits is toggled. For each generation of 40 individuals, each being a 45 bits string, a total of 1800 bits are scanned for mutation. At a mutation rate of 0.001, on average just under 2 bits will be toggled per generation.

After all 20 pairs of parents have gone through crossover and mutation, 40 children are generated. The parents are then discarded while their children constitutes a new generation. Each new member of the population is then assessed and the cycle of parent selection according to performance and reproduction repeats. The GA cycle is summarised in figure 7.6.

7.6 Results and Conclusions

7.6.1 Potting Accuracy Assessment

At this point it is appropriate to give an assessment of the intelligent robot snooker player's ability to pot balls. It must be remembered that the game of snooker is about potting ball and scoring points. No amount of intelligent game planning is of any good if the system fails to pot balls. Hence a series of tests were carried out to assess the performance of the robot snooker player.

The straight line potting accuracy of the system is first assessed. A straight line pot is one where the pocket, target ball, and the cue ball form a straight line. The tests are set up such that the target ball and the cue ball always lie on the snooker table diagonal. The

difficulty can be altered by varying d, the distance between the pocket and the target ball (which is also the distance between the target ball and cue ball). The value of d was set to 350 mm, 500 mm, and 650 mm, which correspond to short, medium, and long range shots. Each shot was performed 10 times by the robot so that any irregularity in the results can be spotted. Illustrations of the test shots and results can be found in Figure 7.7. Each of the 30 test shots was successful.

In actual game playing, angled shots are more frequently encountered. In an angled shot, the line joining the pocket point and the target ball makes a non-zero angle θ with the line joining the target ball and the cue ball. In this asssessment, the distance between the pocket point and the target ball, and that between the target ball and cue ball are maintained at 400mm. The trial shots were set up such that the line joining the pocket point and the target ball bisects a corner pocket. Five trial shots were set up where the value of the cut angle equals 10°, 20°, 30°, 40°, and 50° respectively. Again, each trial shot was played 10 times by the robot. Results can be found in Figure 7.8. Note that the success rate of angled potting is 100 % up to a 40° cut. For the 50° cut, however, the robot failed all 10 attempts. Observation of the shots being taken shows that the robot always miss by the same margin, which indicates good robot repeatability but insufficient accuracy.

7.6.2 Genetic Algorithms Performance Assessment

The rate at which convergence to the optimal occurs is a good guide to the effectiveness of genetic algorithms. In problems where numerous local optimals exist, GA may be confused thus missing the global optimal. Indeed the basic implementation of GA as described earlier exhibits premature convergence. In this section, the probable causes of premature convergence are examined and remedies put forward, which will be backed up by performance analyses. Apart from the basic implementation of GA, 3 other variations incorporating different modifications are experimented upon.

Owing to the random nature of genetic algorithms, it is inappropriate to compare performances using performance figures from a single trial run of a certain number of generations. Instead, each version of GA is allowed to run up to 1000 generations 20 times. Each trial run begins with a different random initial population. The fitness values of the best individual at each generation of the 20 trial runs are averaged. A *convergence curve* is then produced by plotting the average best fitness against the generation number. The convergence curve thus produced gives a much better indication of the rate of convergence towards the optimal.

7.6.2.1 GA1 - the basic genetic algorithm

GA1 is the basic GA program written in QuickBasic. Figure 7.9 shows the convergence curve of GA1. The curve exhibits moderate convergence from generation 0 to 200. But from then onwards, the curve seems to oscillate around fitness value 650. Also, a large number of local peaks and troughs exist on the curve. Further analysis of GA1 reveals that in 200 generations, the maximum number of duplicate individuals equals 37. Bearing in mind the population size of 40 individuals, this equates to only 3 unique individuals within the entire population. On average, the number of duplicates equals 25, more than half the population. Increasing the population size to 100 individuals and repeating the analysis reveals a similarly high proportion of duplicates. Since the power of GA relies heavily on its massively parallel search characteristic, its performance would be severely handicapped if a large number of duplicates exists in the population. Existence of duplicates in the population has the following implications :

(i) Reduced variety of individuals within the population :

As duplicate individuals begin to dominate a population, the probability of the creation of new offspring is reduced thus encouraging local convergence.

(ii) Redundant fitness assessment of duplicates :

Identical individuals have equal fitness. Therefore if the population contains n copies of the same individual, the fitness assessment routine would be invoked redundantly (n-1) times.

(iii) Disrupts the correct functioning of biased roulette wheel parent selection :

This is by far the most damaging aspect of duplicates within a population. In fact biased roulette wheel parent selection fails if duplicate individuals exist. The objective of biased roulette wheel parent selection is to ensure that high performance individuals have proportionately higher chances of being selected to reproduce. In the presence of duplicates, the objective is no longer upheld. Suppose the population contains n copies of an individual I, the corresponding biased roulette wheel will have n compartments associated with individual I. Therefore the probability of individual I being selected is n times higher than if there were only 1 copy of it. This gives the individual an unfair advantage in the contention to

reproduce such that a superior individual with no duplicates may have a lower chance of being selected than an inferior but duplicated one.

7.6.2.2 GA2 - genetic algorithm incorporating duplicates removal

Having identified the problems caused by duplicates, the logical modification to GA1 is to remove duplicates from the population each time a new generation is created. Given a current generation G_n of 40 individuals, a biased roulette wheel can be constructed after every individual's fitness has been assessed. After selection and mating (the collective term for crossover and mutation) of 20 pairs of parents, 40 new individuals are created and stored in an array ' *Offspring* '. In GA1, ' *Offspring* ' becomes G_{n+1}, thus completing a GA cycle. In GA2 the array ' *Offspring* ' is repeatedly ' *compressed* ' and ' *filled-up* ' until no duplicates exists. ' *Compression* ' involves scanning the array ' Offspring ' such that each time a duplicate is detected, it is deleted and the following array members are shifted up. ' *Fill-up* ' generates new individuals using the biased roulette wheel to fill up the vacancies left behind by the deleted duplicates. Only when no duplicate is detected by ' *compression* ' is the array ' Offspring ' assigned to G_{n+1}. Figure 7.10 shows the convergence curves of both GA1 and GA2. The figure shows that GA2 is an improvement over GA1 in terms of both the rate of convergence and the quality of the converged optimal.

7.6.2.3 GA3 - genetic algorithm incorporating fitness windowing

GA3 is a modified version of GA2 in which fitness windowing is applied. The aim is to improve the fairness of parent selection to prevent an individual of high fitness from dominating the parent pool. This is achieved by modifying the construction of the biased roulette wheel given a population of individuals and their fitness. In both GA1 and GA2, each individual is allocated a compartment on the roulette wheel of a size proportional to its fitness. To increase fairness, the individuals are first sorted according to their fitness in ascending order. The first individual in the sorted list is therefore the least fit while the last individual is the fittest. From this point onwards, the actual fitness values of the individuals play no part in the biased roulette wheel selection process.

By associating each individual with a compartment on the roulette wheel of size equal to its ranking in the sorted list of individuals, increased fairness in parent selection can be achieved. In general, the probability of an individual ranked k^{th} in the ordered list of being selected as a parent equals $k \Big/ \sum_{i=1}^{40} i$.

The convergence curves of GA3, GA2, and GA1 are shown in figure 7.11. Comparison between the convergence curves proves interesting in that GA3 exhibits the best initial convergence (notably before reaching generation number 200) but then started to falter. From generation 200 onwards, GA2 continues to converge while GA3 hardly improves any further. Overall, it has to be said that GA3 performs worse than GA2. Nevertheless, GA3 does exhibit a very favourable quality of rapid initial convergence. This leads to the development of GA4 which is an attempt to capitalise on this aspect of GA3.

7.6.2.4 GA4 - steady state genetic algorithm

GA1, GA2, and GA3 evolve from one generation to another by means of generational replacement i.e. individuals of the current generation are completely replaced by offspring resulting from the mating of the selected parents. GA4 uses a different strategy, commonly known as steady state reproduction, to create a new generation. This involves directly transferring the 10 fittest individuals within a generation to the succeeding generation. As in GA3, individuals are first ranked in ascending order of fitness, with the fittest individual at the tail of the list. The last 10 individuals in the list are then directly transferred into the new generation. This leaves 30 vacancies (as the population size is 40) in the new generation to be filled up by offspring resulting from mating of parents selected using the biased roulette wheel, the construction of which is the same as that in GA3. This strategy allows individuals to compete for places in the ' immortal ' subset within the population. Once an individual becomes immortal, it remains in subsequent generations until it get displaced by individuals of higher fitness.

The performance of GA4 is compared to that of GA1, 2, and 3 in figure 7.12. The figure shows that GA4 exhibits the most rapid initial convergence and at the same time managed to converged to the best fitness. GA2 exhibits moderate initial convergence coupled to a good final converged fitness. On the other hand, GA3 exhibits rapid initial convergence but fails to arrive at a good converged fitness. GA4 appears to combine the advantages of GA2 and GA3 without incorporating their drawbacks.

The robustness and effectiveness of genetic algorithms have been widely documented. GAs have been applied to a broad range of problems with favourable results. However, a drawback due to its versatility is that sometimes convergence occurs only after a very large number of generations and many users demand higher performance. In this application, the relationship between GA performance and the way new generations are created has been investigated. Results show that GA performance is sensitive to the method of parent selection and how a generation is updated.

In all of the convergence curves plotted, the best converged fitness values lie just above 570. Remembering that the fitness of an individual is a measure of how well the ball dynamics model matches 5 example shots using the encoded parameters, this fitness equates to a ball motion prediction accuracy of just under 120 mm per example shot, or approximately 3 balls length. This is acceptable allowing for the various factors (see chapter 7.3) which contrive to corrupt the performance of the model. The parameters encoded in the best performing individual in the 20 trial runs of 1000 generations for GA1, GA2, GA3, and GA4 are listed in table 7.3.

7.6.3 Shot Planning Accuracy Assessment

Six game situation were created to assess the system's performance at devising a game plan and the outcome of executing the game plan. The ball motion parameters determined by GA4 as tabulated in table 7.3 are used to run the system. To ensure clarity of the example plans, only a red ball, a black ball, and the cue ball are present on the snooker table. Six example game plans (I) - (VI) were generated by the game planning module of the intelligent robot snooker player. The objective is to pot the red ball using a suitable pneumatic cue force so that the cue ball will end up in a position to pot the black ball.

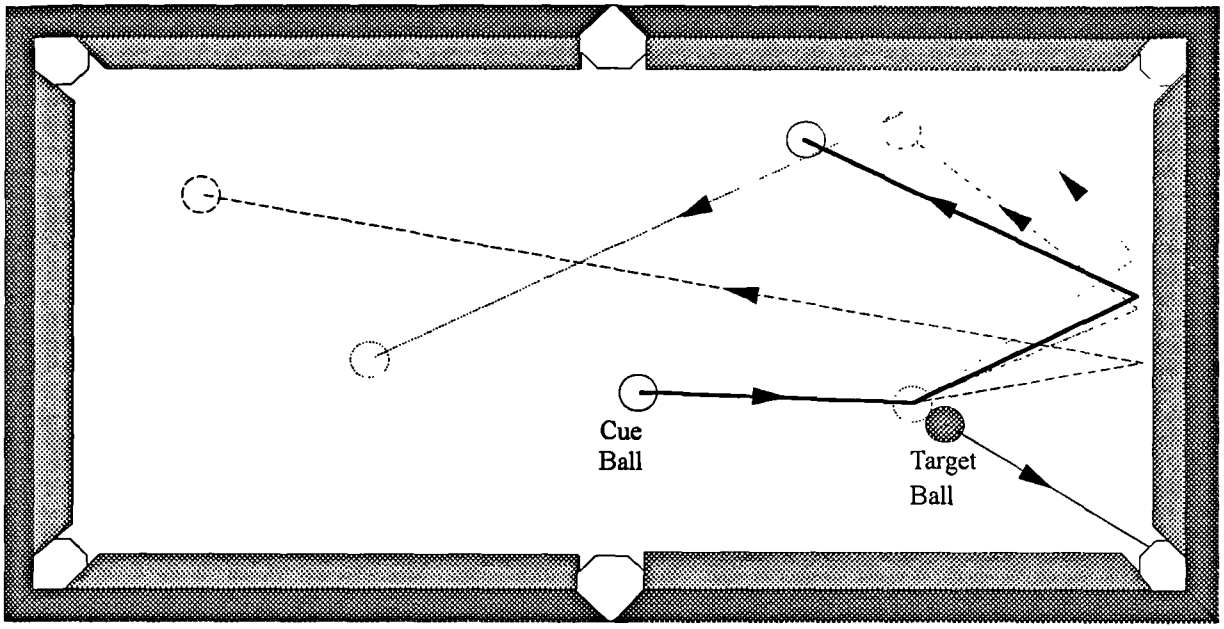
By instructing the robot to play the shot defined in the game plan using the chosen pneumatic cue force, the outcome of the shot, in particular the cue ball path, is noted. It is then possible to compare the predicted cue ball path after potting the red ball to the actual one. The 6 example game situations, the plans generated, and the resulting game situations after potting the red ball are presented in figures 7.13 - 18 respectively. The prediction errors, calculation of which was described in chapter 7.3, are tabulated in table 7.4.

The robot managed to pot the red ball in each of the 6 game plan examples. The real focus of attention here, however, is the correctness of the cue ball path predictions. Figures 7.13 - 18 and table 7.4 show that the system managed to predict the cue ball path after potting the red with varying degrees of success. The prediction made in example game plan (II) was the most accurate with a prediction error of 10.8 cm. The worst prediction was made in example game plan (VI) with an error of 82.9 cm.

Upon closer inspection of example game plans (I) - (VI), it can be seen that immediately after impact with the red ball, a good match exists between the predicted and actual cue ball path. However, as the cue ball travels further, the discrepancy begins to grow. This is most evident in example game plan (VI), see figure 7.18, in which a very high cue force of 232 (maximum cue force = 256) was used to pot the red ball. Fortunately, the

poor performance of example game plan (VI) was the exception rather than the norm. Indeed the average prediction error drops from 27.8 cm to 16.8 cm if example game plan (VI) were not included.

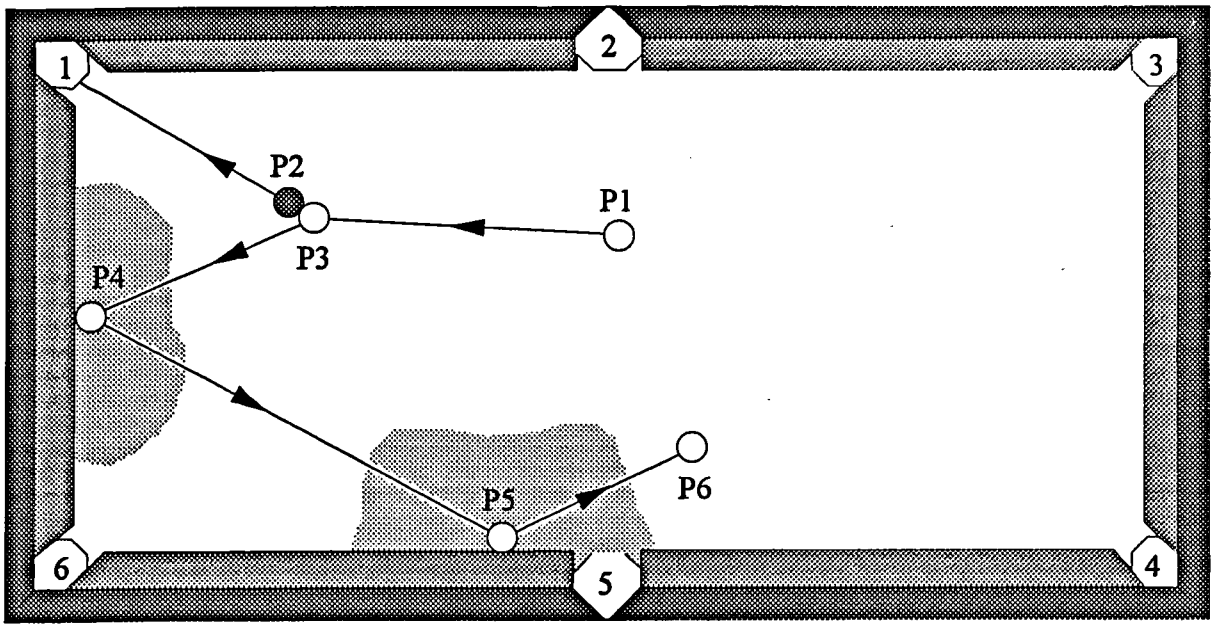
It can be concluded that the present game planning strategy and mathematical model of ball dynamics work with reasonable success. Yet there is no hiding of the fact that the prediction accuracy can be improved. It appears that a critical factor in the accuracy of the cue ball path prediction lies in the modelling of the cue ball to target ball impact interaction. It is possible that the present mathematical model is over simplistic in this aspect e.g. possible rolling behaviour of the cue ball before impact is not taken into account. Frictional forces between the ball surfaces on impact could also have an effect. More research will be conducted in this area to further enhance the performance of the ball dynamics model.



Legend :

- Actual Recorded Cue Ball Path on Playing the Shot
- Predicted Path Using Parameters Set A
- Predicted Path Using Parameters Set B
- Predicted Path Using Parameters Set C

Figure 7.1 Comparison Between Predicted and Actual Cue Ball Paths



Area of snooker table sprinkled with talcum powder to reveal cue ball path

Shot Data : Cue Ball Position = P1

Target Ball Position = P2

Cue Ball Position at Impact with Target Ball = P3

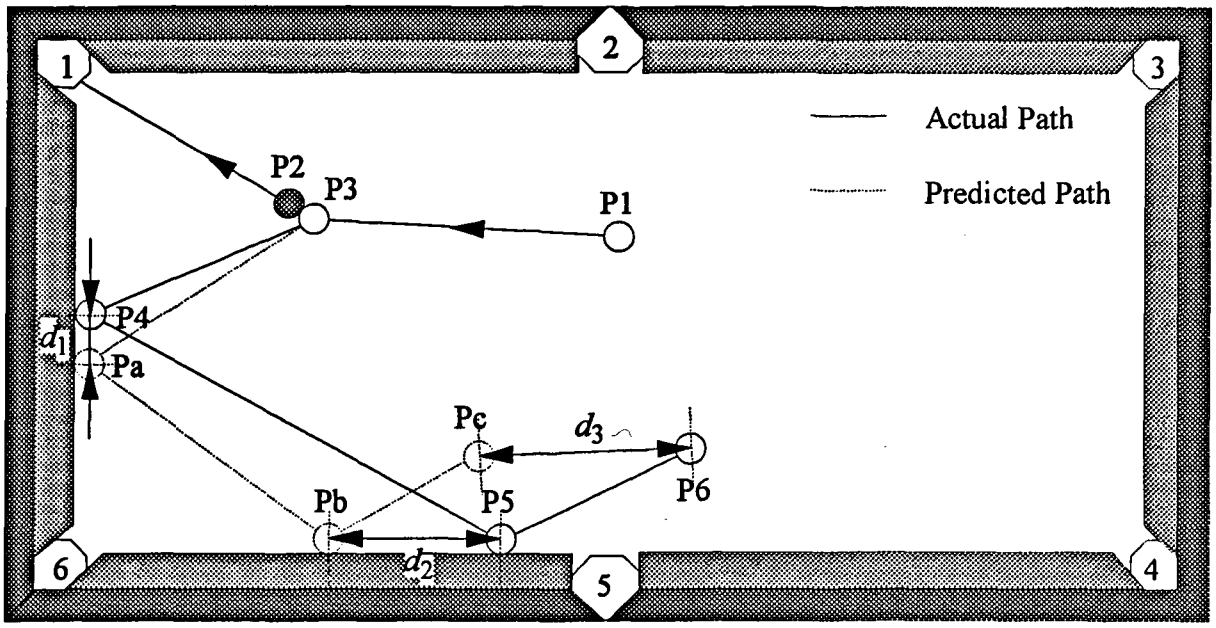
Pocket Number = 1

Cue Force = f

When the shot is played, the cue ball path leaves its 'footprint' on the talcum patches before coming to a rest. This allows the contact points between the cue ball and the cushions to be identified.

By positioning a snooker ball at each of the 2 cushion contact points, and leaving the cue ball finishing position untouched, the vision system is able to determine the co-ordinates of the points P4, P5, and P6, which defines the cue ball path after impact with the target ball.

Figure 7.2 Determination of the Cue Ball Path After Potting a Ball

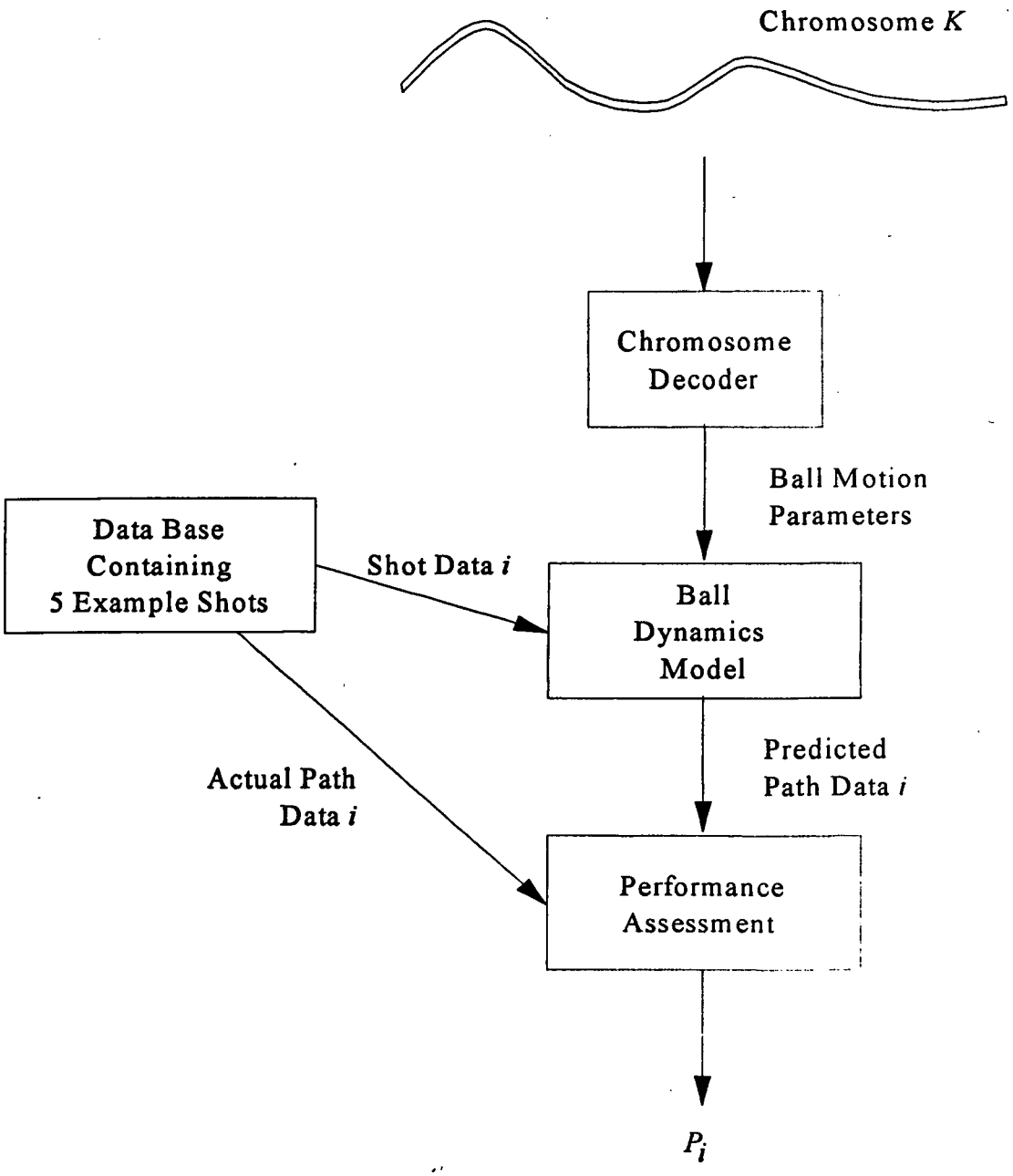


Actual_Path_List = [P4x, P4y, P5x, P5y, P6x, P6y, 0, 0, 0, 0, 0, 0]
Actual_Count = 3

Predicted_Path_List = [Pax, Pay, Pbx, Pby, Pcx, Pcy, 0, 0, 0, 0, 0, 0]
Predicted_Count = 3

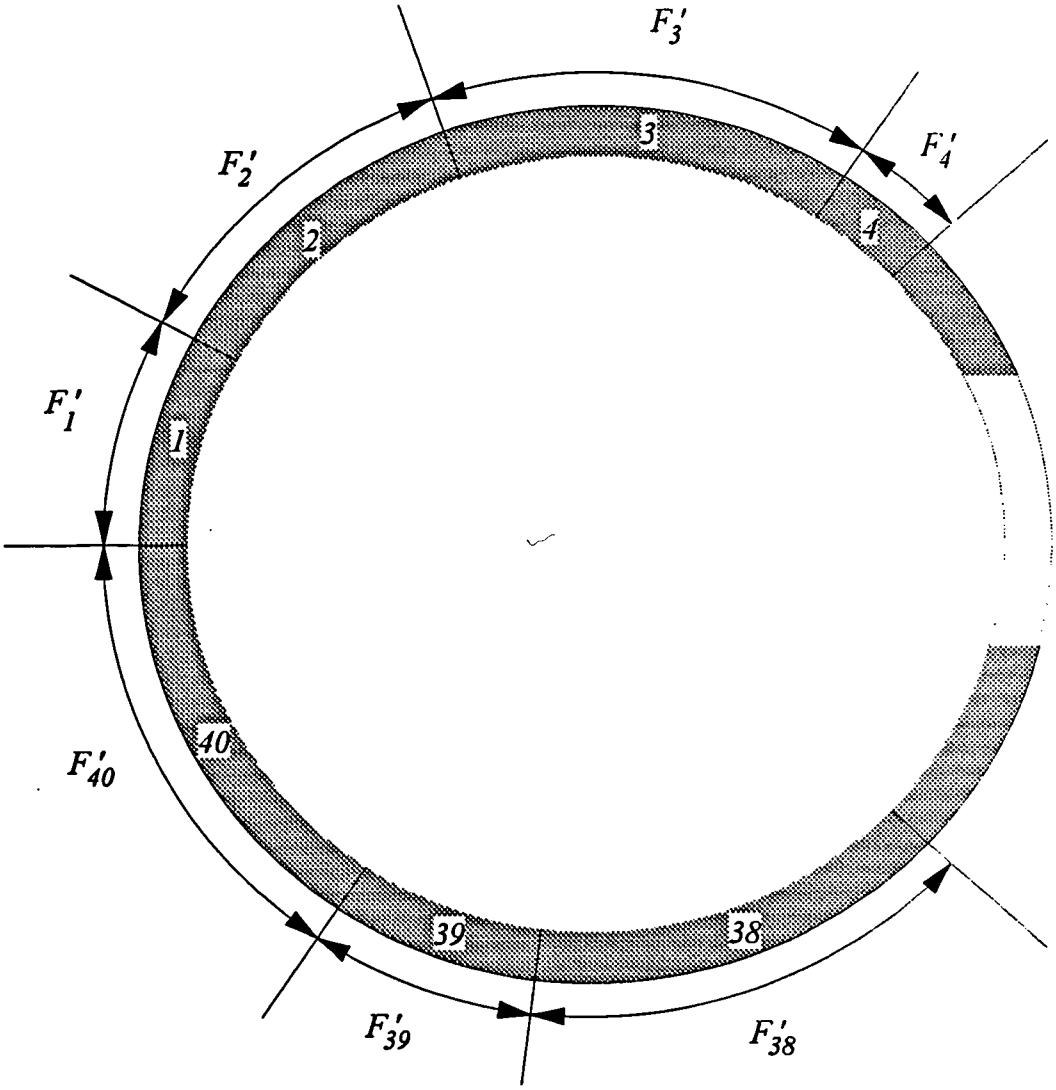
Performance of the prediction = $d_1 + d_2 + d_3$

Figure 7.3 Assessing the Performance of a Predicted Path



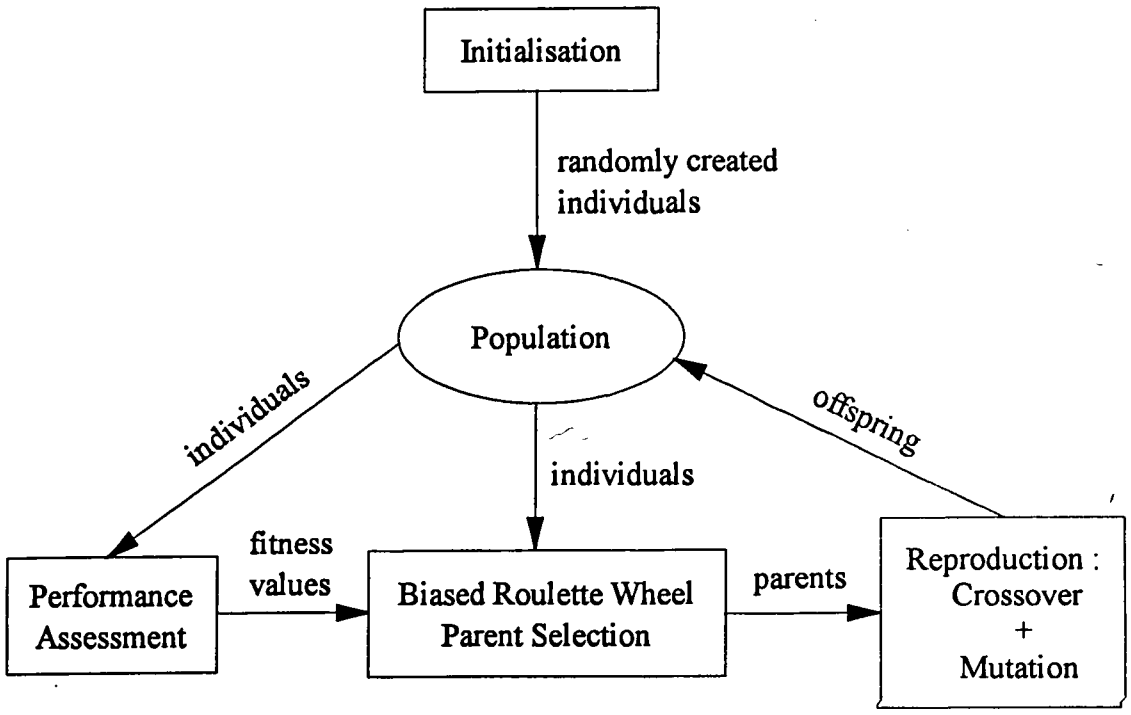
$$\text{Fitness of chromosome } K = F_K = \sum_{i=1}^{5} P_i$$

Figure 7.4 Assessing the Fitness of a Chromosome Within the Population



Note : Compartments 5-37on the wheel are not shown for clarity.

Figure 7.5 Example Biased Roulette Wheel for Parent Selection



Notes :

The GA cycle can (1) repeat indefinitely

(2) stop after a certain number of generations

(3) stop if the fitness of an individual is below a certain threshold

Figure 7.6 A schematic Genetic Algorithm Cycle

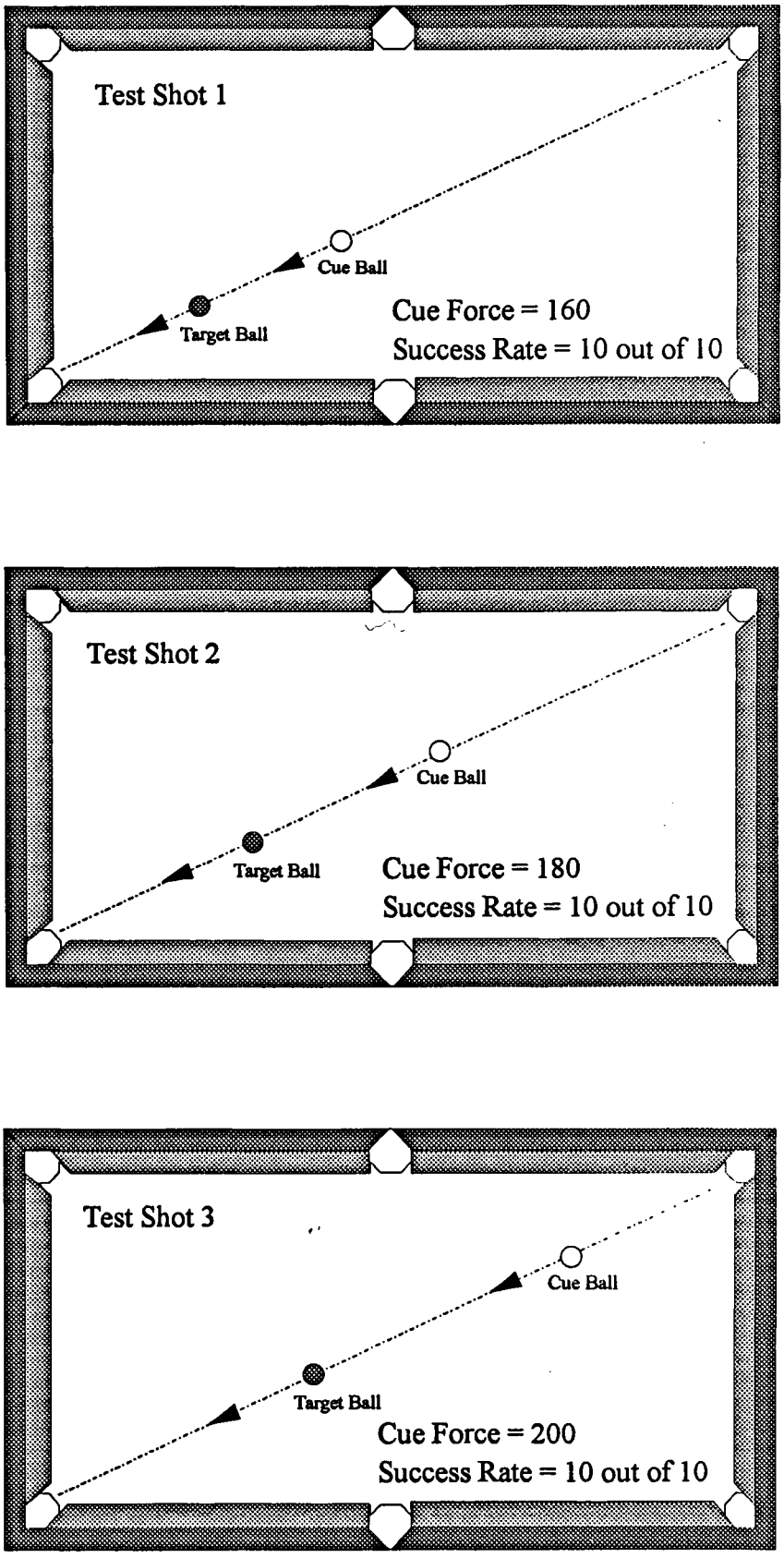


Figure 7.7 Straight Line Potting Tests Results

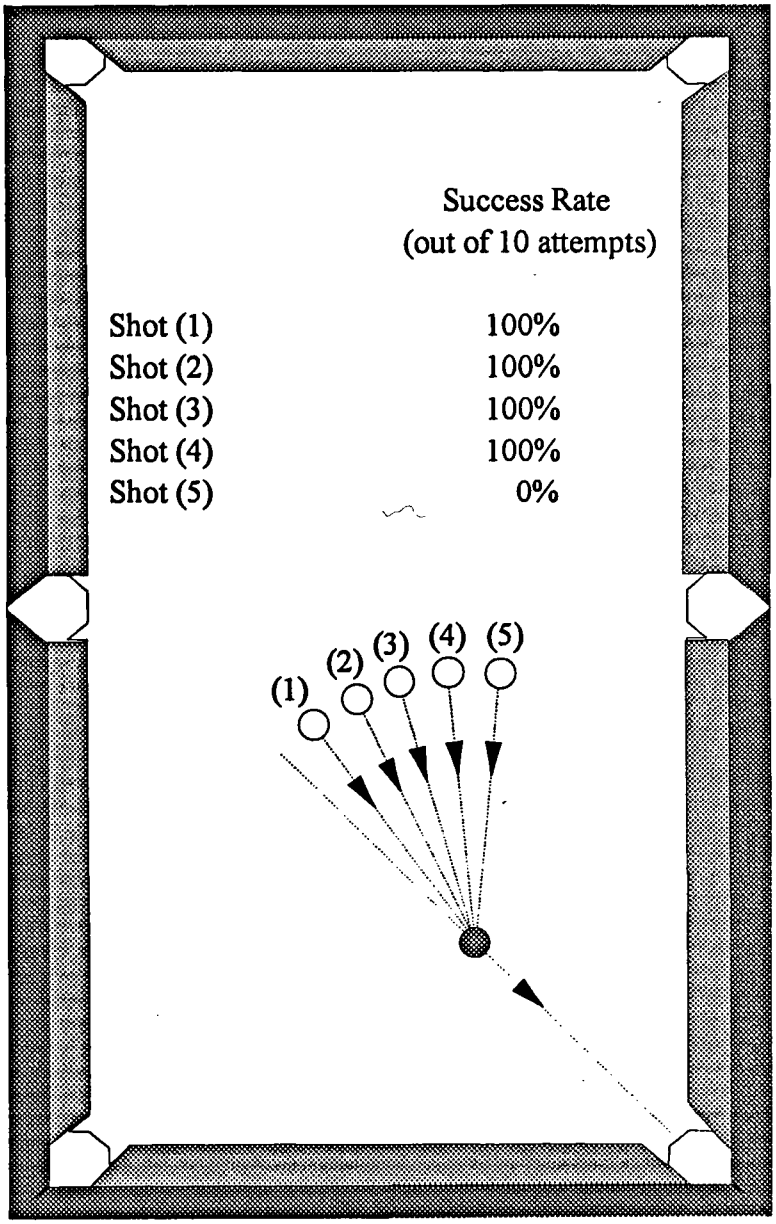


Figure 7.8 Angled Potting Tests Results

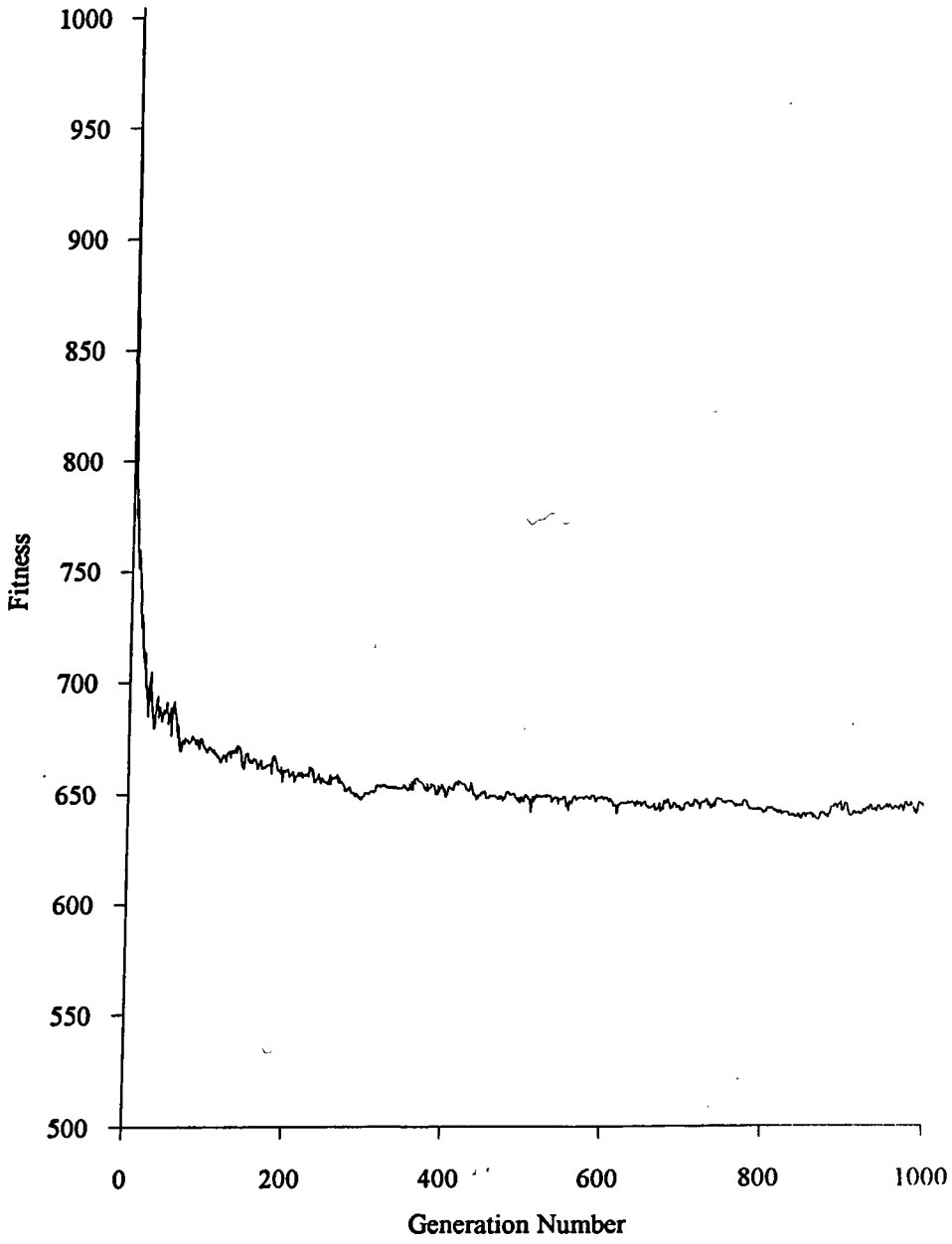


Figure 7.9 GA2 Convergence Curve

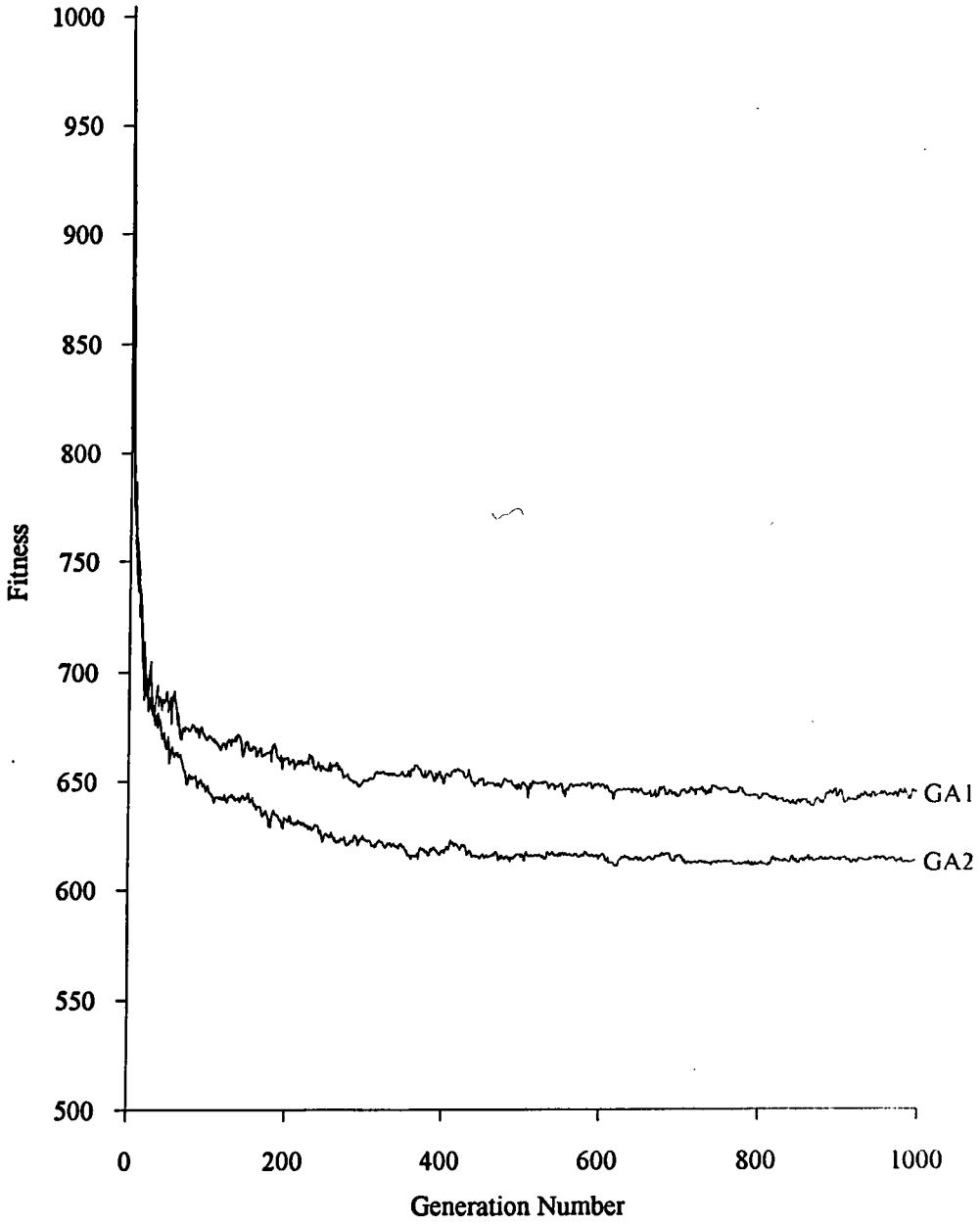


Figure 7.10 GA2 Convergence Curve

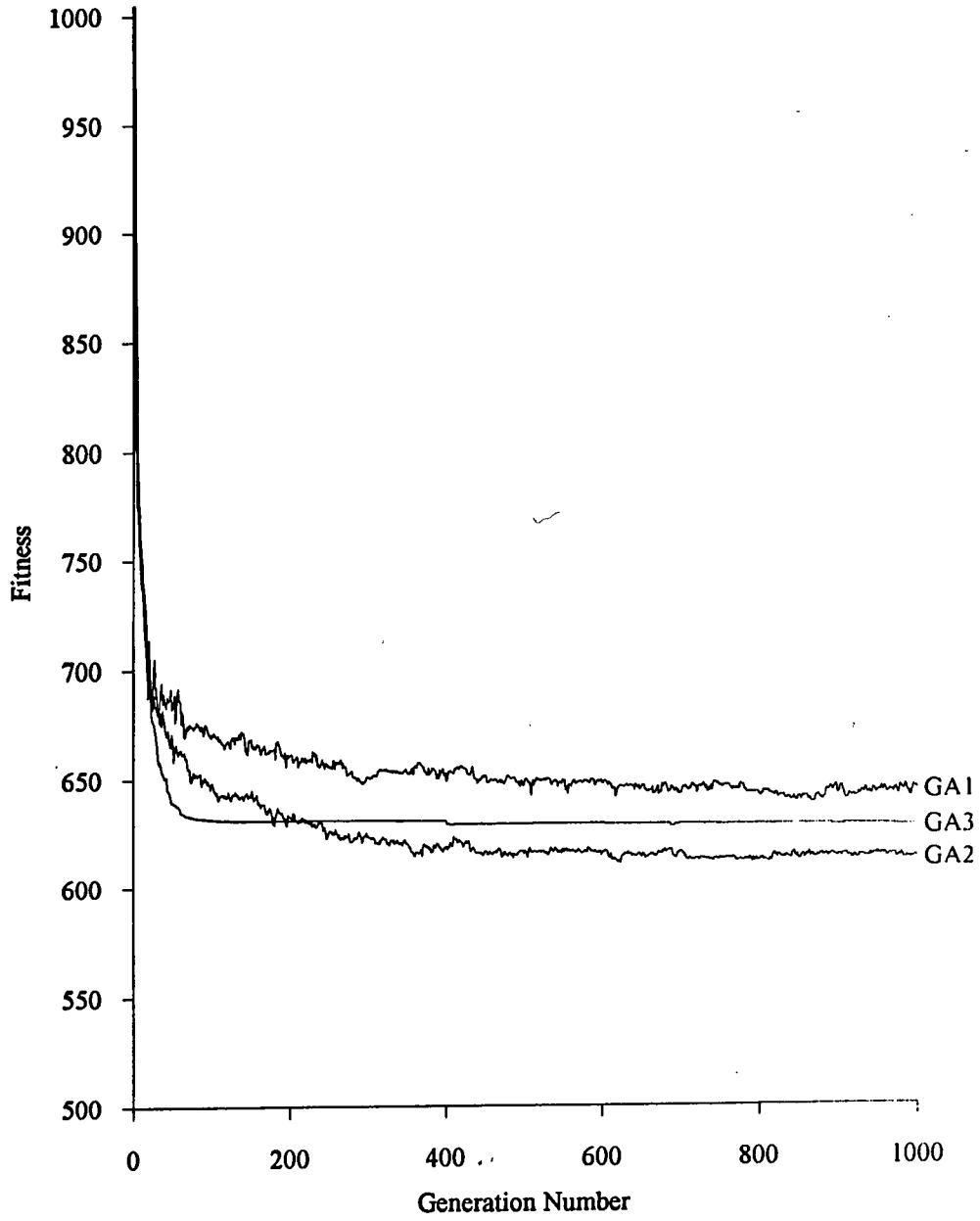


Figure 7.11 GA3 Convergence Curve

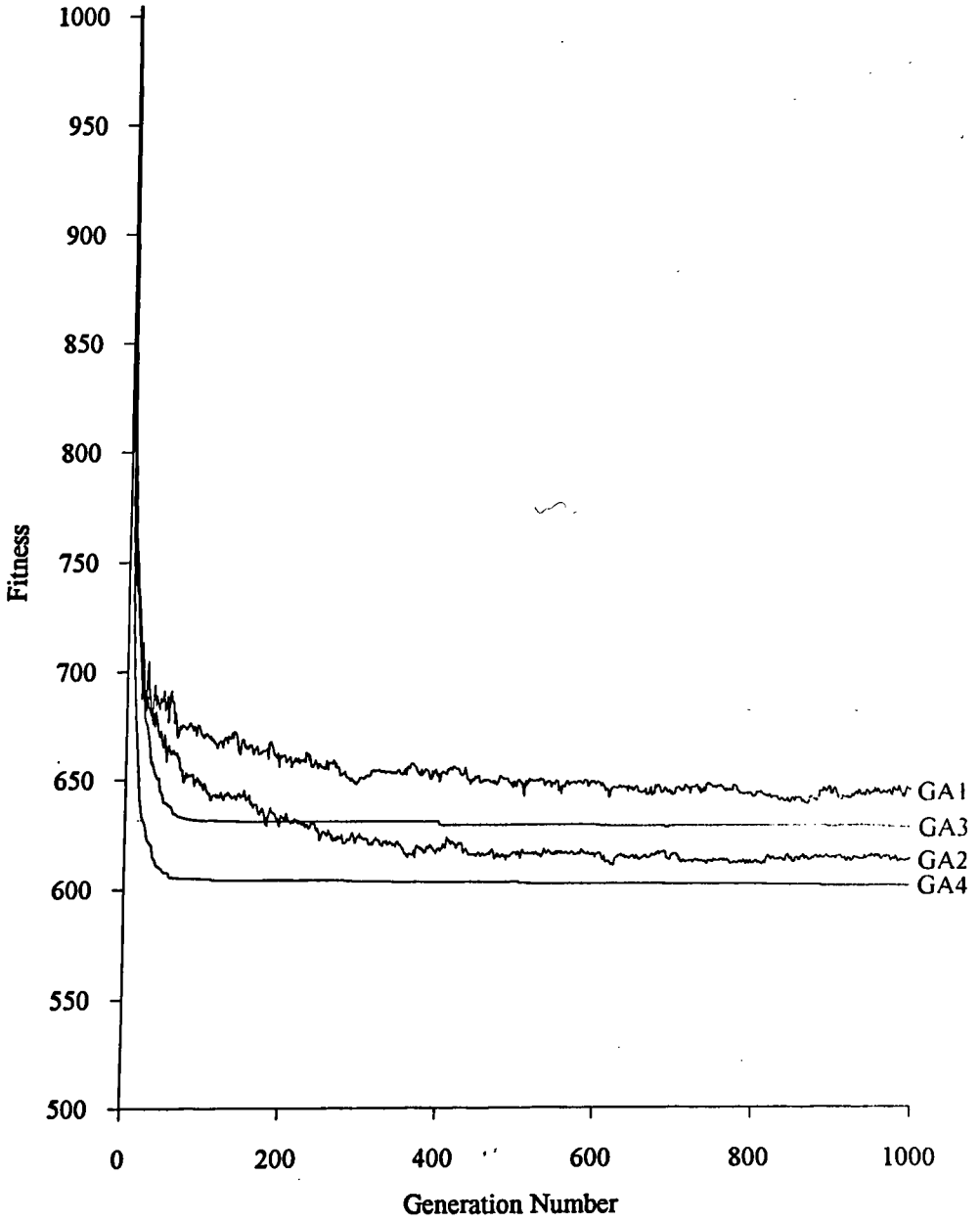
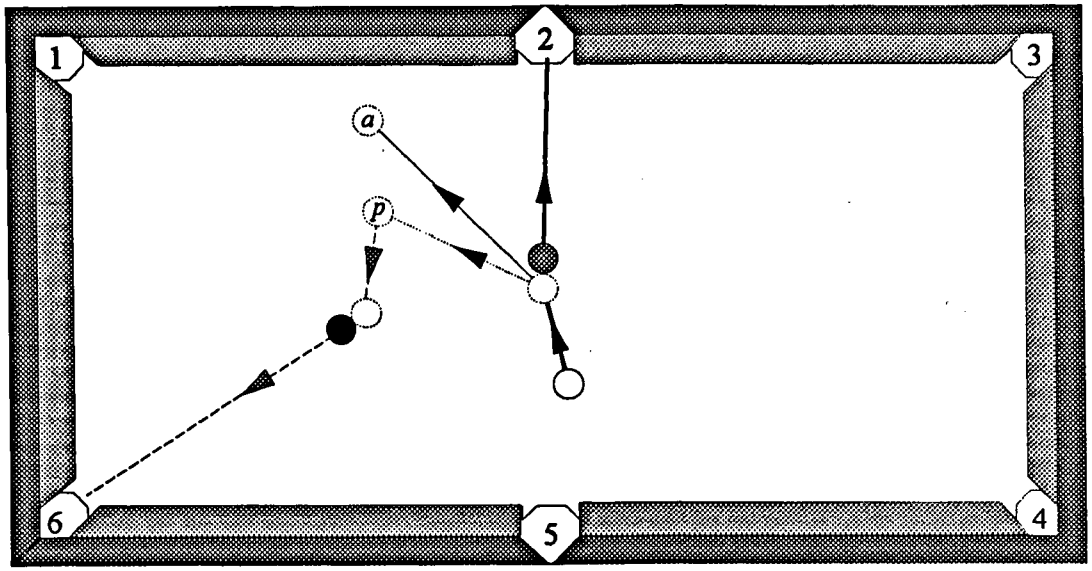


Figure 7.12 GA4 Convergence Curve



10 20 30 cm

cue ball red ball black ball

(p) predicted cue ball position after potting red

(a) actual cue ball position after potting red

First shot played by robot

Planned second shot

Predicted cue ball path after first shot

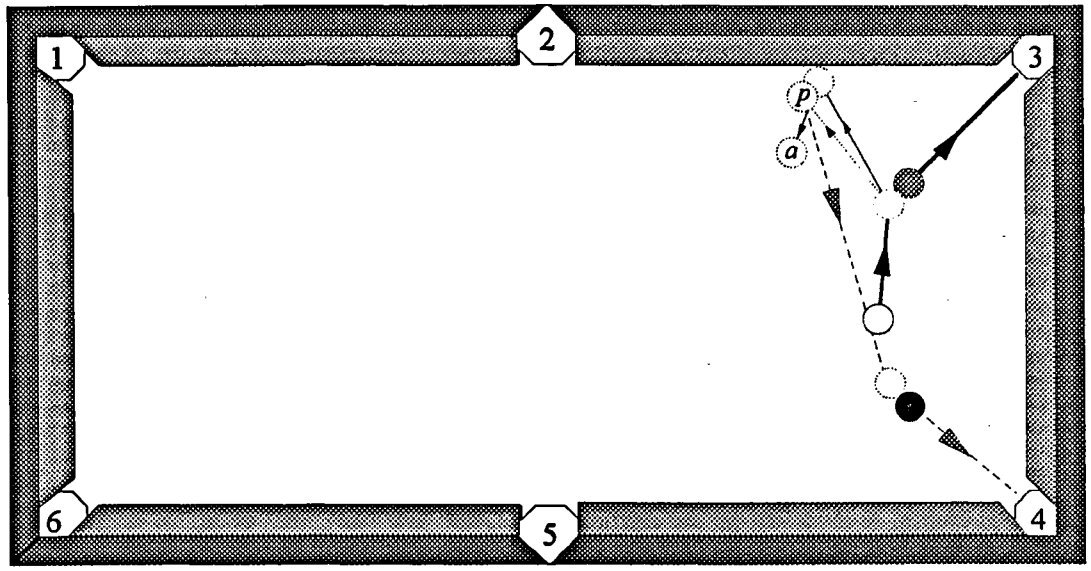
Actual cue ball path after first shot

Game Plan Generated by the Intelligent Robot Snooker Player :

First Shot : To pot red ball into pocket 2 using cue force 209

Second shot : To pot black ball into pocket 6

Figure 7.13 Example Game Plan (I)



10 20 30 cm

cue ball red ball black ball

(p) predicted cue ball position after potting red

(a) actual cue ball position after potting red

First shot played by robot

Planned second shot

Predicted cue ball path after first shot

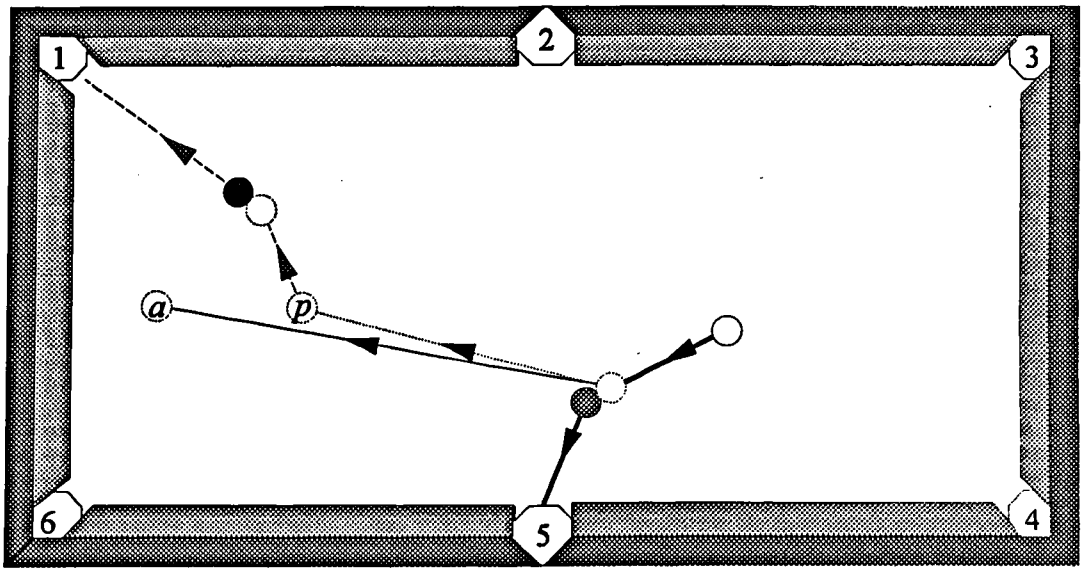
Actual cue ball path after first shot

Game Plan Generated by the Intelligent Robot Snooker Player :

First Shot : To pot red ball into pocket 3 using cue force 53

Second shot : To pot black ball into pocket 4

Figure 7.14 Example Game Plan (II)



10 20 30 cm

cue ball red ball black ball

p predicted cue ball position after potting red

a actual cue ball position after potting red

First shot played by robot

Planned second shot

Predicted cue ball path after first shot

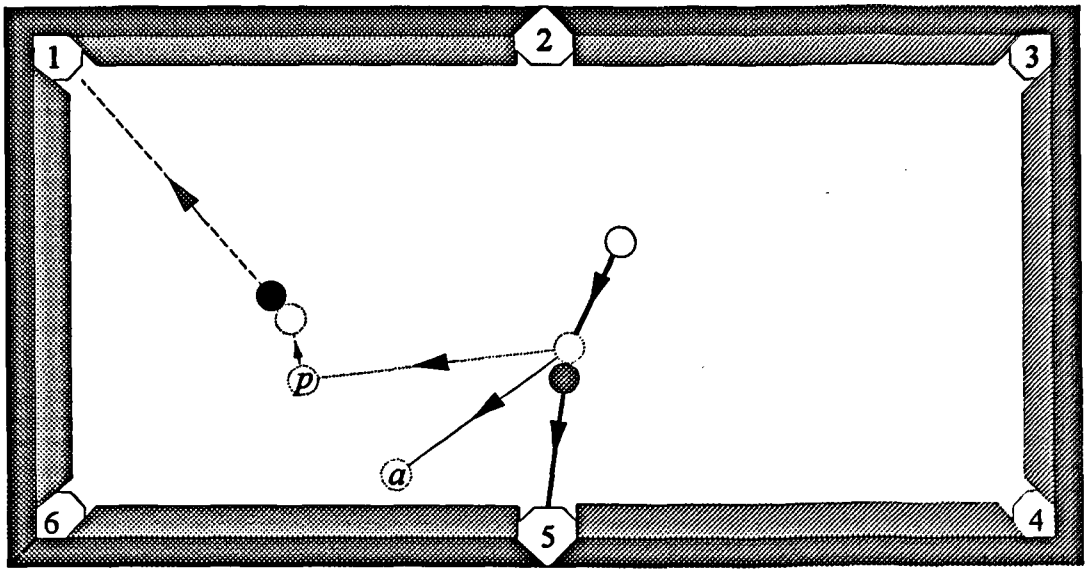
Actual cue ball path after first shot

Game Plan Generated by the Intelligent Robot Snooker Player :

First Shot : To pot red ball into pocket 5 using cue force 62

Second shot : To pot black ball into pocket 1

Figure 7.15 Example Game Plan (III)



10 20 30 cm

cue ball red ball black ball

p predicted cue ball position after potting red

a actual cue ball position after potting red

First shot played by robot

Planned second shot

Predicted cue ball path after first shot

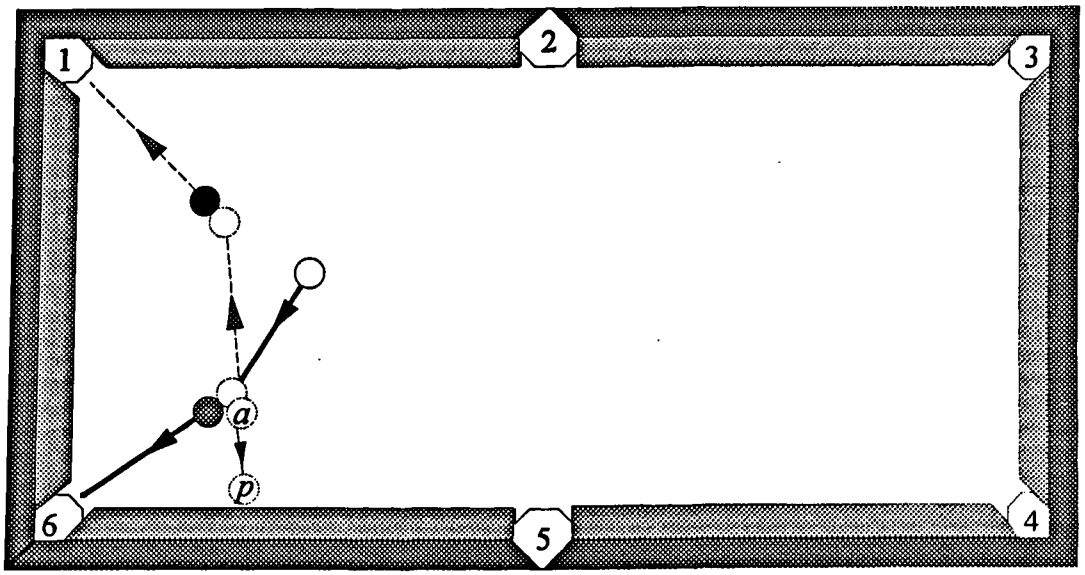
Actual cue ball path after first shot

Game Plan Generated by the Intelligent Robot Snooker Player :

First Shot : To pot red ball into pocket 5 using cue force 111

Second shot : To pot black ball into pocket 1

Figure 7.16 Example Game Plan (IV)



10 20 30 cm

cue ball red ball black ball

p predicted cue ball position after potting red

a actual cue ball position after potting red

First shot played by robot

Planned second shot

Predicted cue ball path after first shot

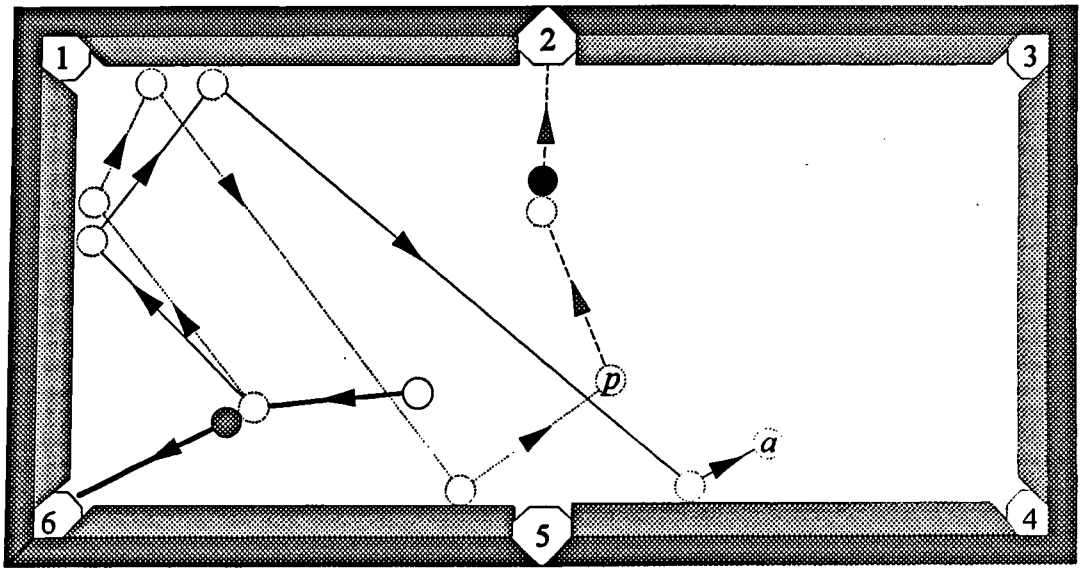
Actual cue ball path after first shot

Game Plan Generated by the Intelligent Robot Snooker Player :

First Shot : To pot red ball into pocket 6 using cue force 49

Second shot : To pot black ball into pocket 1

Figure 7.17 Example Game Plan (V)



10 20 30 cm

cue ball red ball black ball

p predicted cue ball position after potting red

a actual cue ball position after potting red

First shot played by robot

Planned second shot

Predicted cue ball path after first shot

Actual cue ball path after first shot

Game Plan Generated by the Intelligent Robot Snooker Player :

First Shot : To pot red ball into pocket 6 using cue force 232

Second shot : To pot black ball into pocket 2

Figure 7.18 Example Game Plan (VI)

Motion Parameter	Minimum Value	Maximum Value	No. of Bits Allocated for Binary Representation	Precision of Binary Representation
Cue_Ball_Vmax	2000	5000	15	0.09
Mu	0.05	0.5	6	0.007
Omega	50	400	12	0.08
Rest_BB	0.5	1.0	6	0.008
Rest_BC	0.5	1.0	6	0.008

The resultant chromosomal representation of the above parameters is therefore a binary string of length 45 bits.

Table 7.1 Binary Representation of the Ball Motion Parameters

Individual no.	Chromosomes
1	10101111001111010101001000101001001
2	10101101010111111101010011010110101
3	01110111010101001011101010101011010
4	01010110101101011010111111110101110
5	11010101011010000001011010111010101
6	01001010010000110110101111111010001
7	11010101101101011011011011010110110
8	01011010110101110111001010000011111
9	10101010111110100101101001101011010
10	01010010111010100010101011010101111
11	01001101011110110101011100110110100
12	11010110110110101011010110101101101
13	01010011011111010110110101101011010
14	10110101011010110110010110101111101
15	01010110001010101101011010110111101
16	11010101011010000010101101101011010
17	10101101111100010100101110101011011
18	10101100001101010100000000111100110
19	11111110100000000101110101000100011 1
20	10101010111110000000101010100000110
21	10011010101000000010010001100001101
22	00000011111010101001001001111110011
23	10101000111110000001110101010101000
24	11110100101010101001000011010010100
25	01001010001000001011111010010010100
26	10101010010000101010000010100101000
27	00010101010011100100001010100100010
28	10100101001000010110101110100001111
29	10100000100101101010010001010010101
30	00011100101010010001111100101000100
31	11101001010100000101111010100101011
32	00011110100111100101010101010100100
33	01010101010101001010100000000111101
34	01010100100000101001010100001010101
35	11110010011110010111110011110101010
36	10100100100001010000000001110100101
37	11001100110101010100100011110101001
38	11111111100101010010010101001111011
39	00000011101011110010010010100001111
40	00101101010101010101000111111101011

Table 7.2 A Sample Population of Chromosomes

	GA1	GA2	GA3	GA4
Best Individual created in	Trial no. 2 at Generation no. 996	Trial no. 3 at Generation no. 144	Trial no. 17 at Generation no. 51	Trial no. 16 at Generation no. 997
Best Fitness	575.67	578.33	580.39	574.89
Cue_Ball_Vmax (mm /s)	3392.93	3314.74	3632.16	3611.10
Mu	0.05	0.05	0.05	0.05
Omega (mm/s^2)	315.21	293.42	355.90	344.46
Rest_BB	0.86	0.85	0.88	0.87
Rest_BC	0.61	0.60	0.60	0.60

Table 7.3 Best Performing Individuals and Their Encoded Parameter Values

	Prediction Error (cm)
Example Game Plan (I)	15.1
Example Game Plan (II)	10.8
Example Game Plan (III)	23.2
Example Game Plan (IV)	22.5
Example Game Plan (V)	12.5
Example Game Plan (VI)	82.9
Average :	27.8

Table 7.4 Prediction Error of Example Game Plans (I) - (VI)

Chapter 8 Conclusions

The design and implementation of an intelligent robot snooker player has been presented in this thesis. The game of snooker requires a high level of human skills in the potting of snooker balls and in the planning of a game strategy. The aim of this research work is to build a system capable of playing snooker at a level comparable to a good human player, thereby illustrating the possibility of automation of human skilled tasks. The building blocks of the system are conventional equipment such as a black and white image processor, an anthropomorphic industrial robot arm, and a pneumatic cylinder. By integrating them into a functional entity together with the appropriate electronics and computer software, an intelligent robot snooker player has been successfully built.

For the system to play snooker autonomously, it must possess the ability to see, to take action, and most important of all, to make decisions. Indeed these are the prerequisites of many other skilled tasks. Machine vision allows the system to visually analyse an image of the snooker table and its contents. This is accomplished through the use of two CCD cameras : one is located above the snooker table to capture an image of it and the other is attached to the robot arm for close-up positional inspection of a snooker ball. Reliable colour information extraction from a gray-scale image was achieved by analysing the gray-scale maps of the various colours. An eye-in-hand visual servoing mechanism was implemented to facilitate the determination of accurate ball position. The resulting vision arrangement is extremely versatile and can be readily adapted for use in other tasks.

The PUMA robot arm is capable of emulating the human cueing action of stroking a snooker cue to hit a ball. This cueing action was not adopted because of safety and robot kinematic restraints. Instead, a pneumatic cylinder has been converted into a snooker cue which is fitted to the robot end-effector. A good cueing action is achieved by actuating the pneumatic cylinder from the piston-in to the piston-out position. Cue force control is achieved by electronically regulating the air pressures at the inlet and outlet ports of the cylinder.

The PUMA robot arm serves two major purposes in the system : to perform eye-in-hand visual servoing and to position the pneumatic cue to take a shot. The dimension of the snooker table far exceeds the dimension of the PUMA robot arm. By analysing the robot work envelope and using the SKF linear driver as a slave device, the effective work envelope was extended to cover the snooker table. This PUMA-SKF arrangement is applicable to a work area of different dimensions.

The system manages to emulate the thought process of a human snooker player to identify feasible shots, to assess shot difficulty, and to devise a game plan. The imaginary ball technique, together with a series of geometric tests enables the system to decide whether a snooker ball can be potted. This process simulates the way a human player looks along the path of the cue ball, for example, to check for obstructions. It is also important for the system to objectively assess shot difficulty so that the likelihood of success of different shots can be compared. Distances and angles relating to the potting of a ball were used in the computation of a shot difficulty index, the value of which is inversely proportional to the probability of success. A special feature of this assessment is the inclusion of the human player's idea of a preferred angle of cut i.e. the angle between the cue ball and target ball paths in a shot. A preferred angle of cut of 30° was programmed into the system.

Given a random snooker game situation, a large range of actions can be taken. It is possible that a number of target balls can be potted, and each of them may be potted into more than 1 pocket. Furthermore, each feasible shot can be played with a range of cue forces, which shows the complexity of the decision making process. The problem is for the system to find the best course of action amongst the numerous possibilities. A mathematical model of the dynamic behaviour of snooker balls and the concept of snooker game tree have been developed to solve the problem.

Essentially, the mathematical model allows the resulting game situation of potting a ball using a certain cue force to be predicted before the shot is played. This involves modelling the behaviour of the cue ball from the moment it was hit by the cue to the moment its motion stops. This modelling takes into account factors such as frictional losses and momentum transfer. The mathematical model will work accurately only if the values of the various motion parameters were correctly determined.

A technique known as Genetic Algorithms has been successfully utilised in the determination of the relevant parameters required in the modelling of snooker ball behaviour. Genetic Algorithms is an increasingly prominent technique in optimisation and artificial intelligence. A novel method has been devised to compare the expected cue ball trajectory according to the dynamics model with the actual cue ball trajectory through the use of example shots. Genetic Algorithms allows the solution space to be efficiently searched to find the optimum. The performance of the bare basic genetic algorithm has been compared to modified versions and the results are illuminating for other users of GAs. It has been established that a key feature of the algorithm is the maintenance of a population of unique solutions. Performance analysis of the various versions implemented shows that the presence of duplicated solutions in the population

significantly reduces both the performance in terms of rate of convergence and the quality of the converged optimal.

The concept of snooker game tree enables the structured and organised search of a game situation to generate a 1-step ahead game plan. The current game situation is considered the parent node of the game tree. The mathematical model allows the game situation resulting from each feasible to be predicted, thus forming the child nodes of the game tree. The (target ball, pocket, cue force) tuple defines the first shot of the game plan. The cost of transformation from the parent to a child node is equated to the shot difficulty index. Within each child node, a search for the second shot associated with the minimum difficulty index is carried out. The objective is to determine the 1-step ahead game plan where the combined shot difficulty index of the first and second shots are minimal. Once a game plan is generated the robot will be instructed to physically play the shot specified in the plan.

The intelligent robot snooker player system plays snooker in accordance with the rules of the game. The system can also deal with unusual game situations and take appropriate actions. In situations where no ball can be potted, the system will play defensively by hitting the cue ball so that it just manages to touch a target ball, using a snooker table cushion to deflect the cue ball towards the target ball when necessary. The system can play against a human opponent fully autonomously.

In the game of snooker, skill is required both in the planning process and the shot taking process. In other games such as chess, if for example the game plan demands the moving of a pawn forward by one square, then this move can be achieved without any ambiguity or risk of failure. In snooker, there is no guarantee that the game plan can always be adhered to due to failure to pot a ball, a foul being committed, or the incorrect prediction of cue ball position. Hence snooker can be viewed as a form of mechanical chess game in which planning a move and physically making a move are equally important towards success. Test results on the system's performance in potting balls and game planning discussed in chapter 7 show good system performance. The research objective was met and the system satisfies the definition of skilled robots put forward by Khodabandehloo (see page 2).

Finally, the intelligent robot snooker player has successfully demonstrated the feasibility of robotic application in skilled tasks. The possibility to create an artificial system to replicate human skills and to achieve similar capability to a person (in playing snooker in this instance) has been successfully confirmed. This achievement is all the more remarkable bearing in mind the low-cost nature of the system and the use of conventionl equipment as building blocks. At the current pace of developments in alternative,

creative, and skilled robotic applications, the benefits of automation are destined to spread ever wider.

Chapter 9 Future Work

The intelligent robot snooker player system can be improved in several areas. The overhead lighting grid performed as expected in most situations. In general moderate variation in the ambient lighting has no discernable adverse effect and ball colour information is always correctly derived from the digitised image. Problem is experienced in bright summer days and when the sun is low. In such situations, the sun shines almost directly into the laboratory, thus increasing the overall illumination level received by the overhead CCD camera. This results in all colours appearing brighter than expected and the system may wrongly identify the yellow ball as the cue ball, for example. The effect of extreme variations in ambient lighting can be reduced by electronically regulating the overall brightness of the digitised image.

Another vision related problem is that the system cannot distinguish whether two snooker balls are touching. This a indeed a good illustration of a seemingly trivial task that proves difficult to automate. To tell whether two balls are touching demands a machine vision accuracy that far exceeds the resolution of the on-board camera which is no better than 0.1 mm. The rules of snooker states that if the cue ball is in contact with another ball, then the cue ball must be played away from the touching ball. This problem is so difficult that sometimes even an experienced referee has to look at two very close balls from different angles before it can be decided whether they are touching.

The PUMA 560 type of robot has a nominal repeatability of 0.1 mm. However, in this application, it is the accuracy of the robot arm that has a direct bearing on the successful potting of balls. The PUMA 560 robot arm has already received the Unimation 'Pot-Cal' (Potentiometer Calibration) procedure which is reported to eliminate the majority of joint errors **[Judd90]**. By instructing the robot arm to locate the pneumatic cue around the cue ball at intervals of 15°, the cueing accuracy can be visually inspected. In general the robot manages to cue accurately at the cue ball centre. Yet in certain bearings around the cue ball, deviations of the cue tip from the ball centre of up to 4 mm can be observed. This tends to indicate the presence of both geometric and non-geometric robot errors **[Khalil91], [Kim91], [Renders91], [Veitschegger88]**. A variety of error compensation techniques have been proposed in recent literature on the subject. For example, Chen and Chao **[Chen87]** reported an accuracy improvement from 5.9 mm down to .28 mm after carrying out error compensation on a PUMA robot arm. Zhuang and Roth **[Zhuang92]** also managed to improved the mean positioning accuracy of a PUMA arm to 0.4 mm. In view of these extremely favourable results, robot error compensation will result in significant system improvements.

The performance of the system in terms of the computation time required to generate a game plan can also be improved. In game situations where all 10 red balls and colour balls are presents and that 5 of the red balls can be potted, it takes approximately 2 minutes to generate a game plan. This is acceptable allowing for the complexity of the game tree to be searched and that the Automatix AV-5 image processor doubles as the host computer of the system. A possible way to improve the system performance is to transfer the ball colour and position information to another PC that is dedicated to work out a game plan. Having devised a plan, the new host PC can then instruct the PUMA robot arm (either via the AV-5 or directly) to execute the plan. This should involve a simple translation of codes from RAIL 6.04 to a PC-based language such as QuickBasic, Pascal, or 'C'.

One of the most fascinating aspects of professional snooker play is the way a professional player controls the cue ball motion precisely. To replicate this skill by a robot would require improvements to be made to both the system software and hardware. The mathematical model of snooker ball dynamics has to be modified to take into account the initial rotation of the cue ball resulting from off-centre cueing. The impact between a spinning striking ball and a stationary one will also have to be investigated and the findings will be incorporated into the model. When the cue ball is struck off-centre, a proportion of the cueing force will be converted into angular acceleration of the cue ball, thus reducing the initial momentum of the cue ball. To account for this likely reduction in initial momentum, a more powerful cueing device will be required.

Suppose the system is modified to play with spin and side, a more efficient game planning strategy is likely to be needed. This is because there exists an infinite number of possible cueing points around the cue ball centre, which can give rise to infinite branches in the game tree. It would be impractical to perform an exhaustive search for the best shot. Hence it may be necessary to use some heuristic rules to decide when the cue ball should be hit off-centre. Alternatively, the range of cueing position can be restricted to centre, top-spin, bottom-spin, left-hand-side, and right-hand-side. Further investigation has to be conducted before arriving at a feasible strategy.

The present system can be described as programmed to play aggressively in the sense that the robot always aim to pot balls whenever possible. A human player, however, may choose to play defensively when necessary. Defensive play is about sending the cue ball into strategic position without potting any ball such that a difficult game situation will be presented to the opponent. The most common form of defensive play is to leave the cue play in a position where the opponent cannot hit any target ball directly i.e. snookered. Another form of defensive play is to leave the cue ball touching a cushion so that it is

very difficult to play. Good defensive play actually demands a higher skill level then 1-step ahead positional play. In defensive play, the target ball hit by the cue ball will not be potted. Thus the path of the target ball has to be projected apart from that of the cue ball. A further complication to the problem is that the target ball path may cross the cue ball path, which means that the two ball will collide with each other more than once. It would be very challenging to modify the system so that the robot can decide when and how to play defensively.

Appendix A Generation of PUMA Plane Envelopes

As discussed in chapter 4, the PUMA robot arm is only required to work in two horizontal planes, in which the on-board camera and the pneumatic cue work. The area within reach of the PUMA in a horizontal X-Y plane will be termed the plane envelope. The objective here is to find this plane envelope corresponding to the on-board camera and the pneumatic cue.

Referring to figure 2.5, it can seen that the largest PUMA plane envelope is a circle of radius 0.864 m, which lies in the X-Y plane Z = 0. This plane envelope is that of the PUMA default 'NULL' tool (0, 0, 0, 90, -90, 0), which is defined as the centre point of the PUMA tool mounting flange at the robot wrist and the orientations identical to that of joint 6 of the robot arm. For the on-board camera, the tool transformation 'CAMTOOL' is (-3.25, -51, 0, 90, -90, 0) and 'CUETOOL', the tool transformation for the pneumatic cue is (170, 0, 80, 90, -90, 0). Both tool transformations have the same orientation as the 'NULL' tool. It is important that the appropriate VAL-II tool definition command be issued before the generation of the plane envelope i.e. before generating the on-board camera plane envelope, the command 'TOOL CAMTOOL' must be issued and similarly for the generation of the pneumatic cue plane envelope.

The system is set up such that during visual servoing using the on-board camera, PUMA works in the X-Y plane Z = 276 (all dimensions in mm). And when the pneumatic cue is lined up at a snooker ball on the table horizontally, the PUMA is in the X-Y plane Z = 573. The relative position of these two plane envelopes are shown in figure A.1.

Figure 2.5 and A.1 show that neither the on-board camera plane envelope nor the cue plane envelope can exceed the maximum plane envelope when Z = 0. As a result, it is sufficient to scan the X-Y plane starting at the point (-900, -900) to (900, 900) using a double FOR loop as below :

```
TOOL CAMTOOL {or CUETOOL}
Z = 276 {or 573 for pneumatic cue}
BELOW {or ABOVE}
RIGHTY {or LEFTY}
FOR Y = -900 to 900 STEP 5 DO
        FOR X = -900 to 900 STEP 5 DO
        {check whether the point (X, Y, Z) is a plane envelope boundary point}
        {If YES, record the point}
        END
END
```

Note that the PUMA X-Y plane is scanned at intervals of 5 mm. With a smaller scan interval, the plane envelope boundary can be more accurately traced with more boundary points detected. But as the interval gets smaller, two problems occur : first, the amount of data storage required to store all the boundary points may exceed the system's memory capacity and secondly, the time required to complete the scan can be prohibitively long. On the other hand, too large a scan interval would not produce an accurate trace of the plane envelope. A scan interval of 5 mm is considered to be a reasonable compromise between execution time and the quality of the plane envelope generated.

Also not shown in the above pseudo-code is how to test whether a point (X, Y, Z) marks the plane envelope boundary or not. To test the point (X, Y, Z), a second point (X+5, Y, Z) is also looked at. A general point (X, Y, Z) is either within range of the PUMA tool according to a set of criteria (this criteria is explained in the next paragraph) or it is not. Therefore the two points (X, Y, Z) and (X+5, Y, Z) can have the following status:

(X, Y, Z)	(X+5, Y, Z)	Comment
IN RANGE	IN RANGE	Both points within plane envelope
IN RANGE	**NOT IN RANGE**	**(X, Y, Z) is a boundary point**
NOT IN RANGE	**IN RANGE**	**(X+5, Y, Z) is a boundary point**
NOT IN RANGE	NOT IN RANGE	Neither point within plane envelope

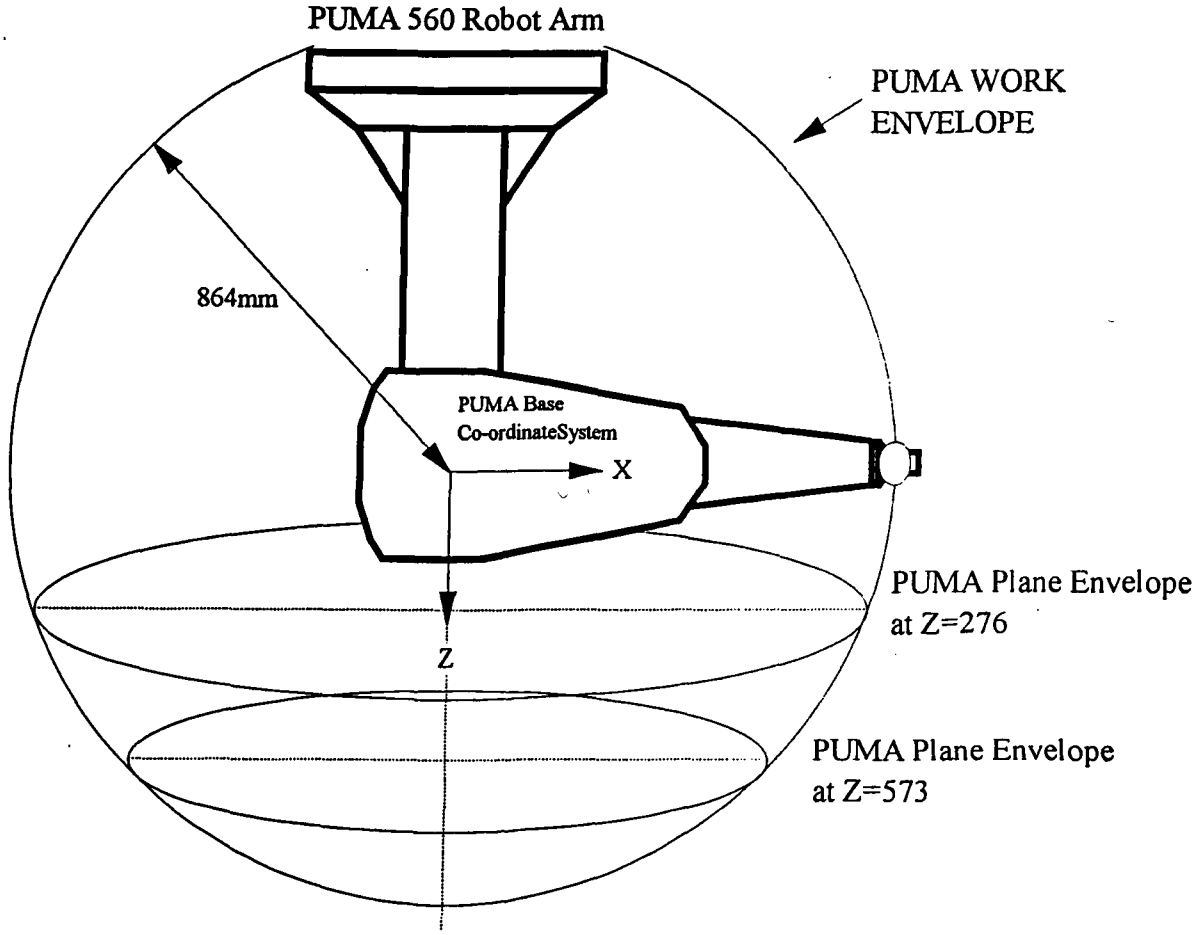
Any boundary point detected according to the above status table is added to an array of boundary points. Note that whether a point (X, Y, Z) is in range or not also depends on the required orientation specified by the transformation parameters (O, A, T). However, the tool orientation varies with the type of operation the PUMA is performing and so is omitted here.

VAL-II provides a command 'INRANGE (TRANS(X, Y, Z, O, A, T))' which returns the value zero if the specified point is within range of the PUMA at the current arm posture. 'TRANS(X, Y, Z, O, A, T)' returns a PUMA location transformation where (X, Y, Z) specifies the PUMA Tool position and (O, A, T) specifies the orientation of the tool. But because of the different characteristics of the on-board camera and pneumatic cue in use, two sets of custom in-range tests have to be coded instead of using the basic VAL-II 'INRANGE' function.

The usage of the on-board camera will be looked at first, during which the PUMA tool configuration is set by the command 'TOOL CAMTOOL'. From chapter 2 it is known that the on-board camera always operates in the orientation (O, A, T) = (-90, -90, 0), and also the camera works in the X-Y plane Z = 276. However, even if a point (X, Y, 276, -90, -90, 0) is tested to be within reach of the PUMA using the 'INRANGE' function, there is no guarantee that the PUMA robot arm can perform visual servoing on a ball successfully. It must be remembered that the position of a snooker ball computed by the overhead camera contains a displacement error of up to 20 mm which is why visual servoing is needed in the first place. Hence the PUMA robot arm has to be able to reach any point inside a circle of diameter 20 mm, lying on the X-Y plane Z = 276 and centred at point (X, Y), in the required orientation (-90, -90, 0). It is possible to scan this circle at a certain interval (e.g. 1 mm) similar to the plane envelope scan described above. To save program run time without compromising the scan accuracy, it is acceptable to generalise that if 8 equally spaced point on the circle are within reached, then any point inside is also within reach of the PUMA. This test is shown graphically in figure A.2.

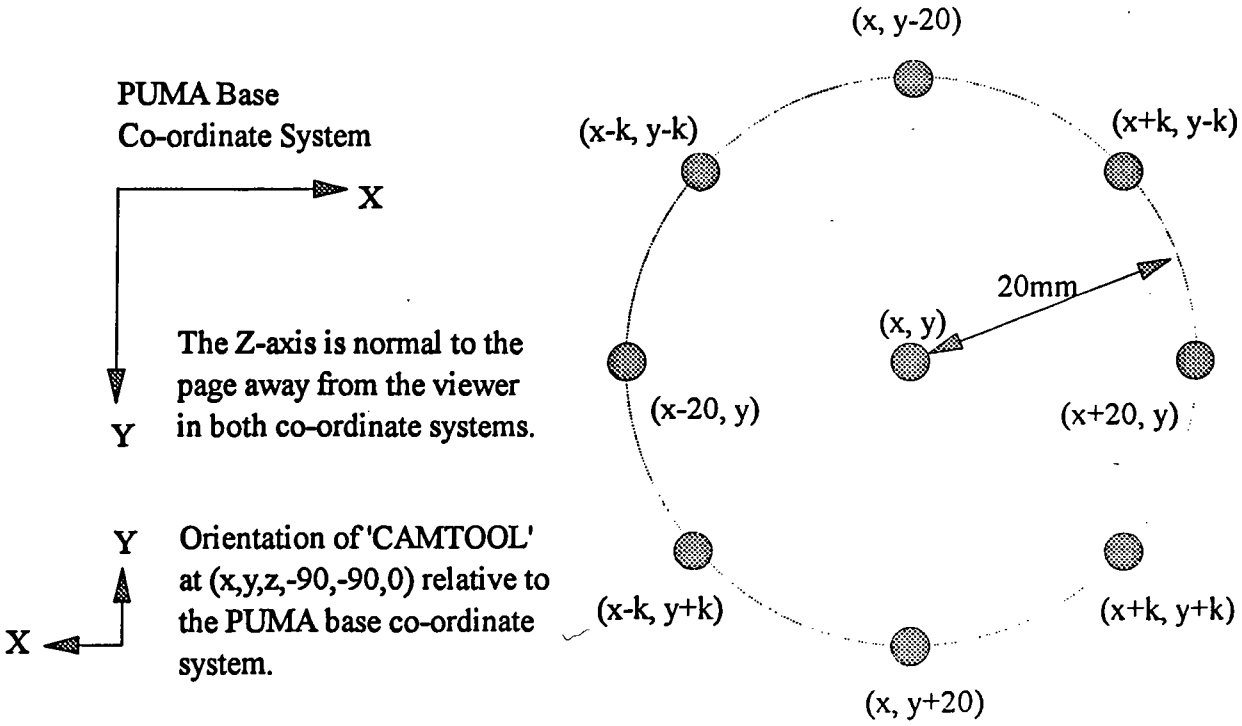
A point (X, Y, 573) is considered to be within reach of the pneumatic cue only if the cue can be positioned at any cue angle at that point with the cue lying horizontally. Admittedly, there are situations where the cue has to be tilted to clear obstacles. However, in those cases, the Z-co-ordinate of the 'NULL' tool point will invariably be decreased as the cue is tilted, and reference to figure 2.5 shows that at a decreased Z-co-ordinate, the plane envelope is bound to increase in size, thus it is not necessary to test if the point is in range with the cue tilted. Again, it can be generalised that if the cue can be positioned at (X, Y, 573) in 8 orientations (at 45° intervals around the point with the cue lying horizontally) then the point (X, Y, 573) is within reach of the cue at any orientation, as seen in figure A.3.

As explained in chapter 2, the PUMA plane envelope depends on the arm configuration, which can be {BELOW, LEFTY}, {BELOW, RIGHTY}, {ABOVE, LEFTY}, or {ABOVE, RIGHTY}. Therefore, for both the on-board camera and the pneumatic cue, the plane envelope generation algorithm has to run four times to find the complete set of plane envelopes. The plane envelope boundary points generated using this algorithm are plotted in figure 2.6 and 2.7.



Note : The Y-axis of the PUMA base co-ordinate system is normal to the page towards the viewer.

Figure A.1 PUMA Work Envelope and Plane Envelopes



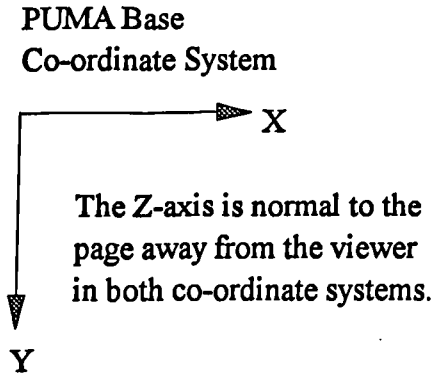
Note : 1.The On-board Camera always operates in the (-90,-90,0) orientation.
2. The On-board Camera always operates in the X-Y plane Z=276
3. $k = 20\cos(\pi/4)$

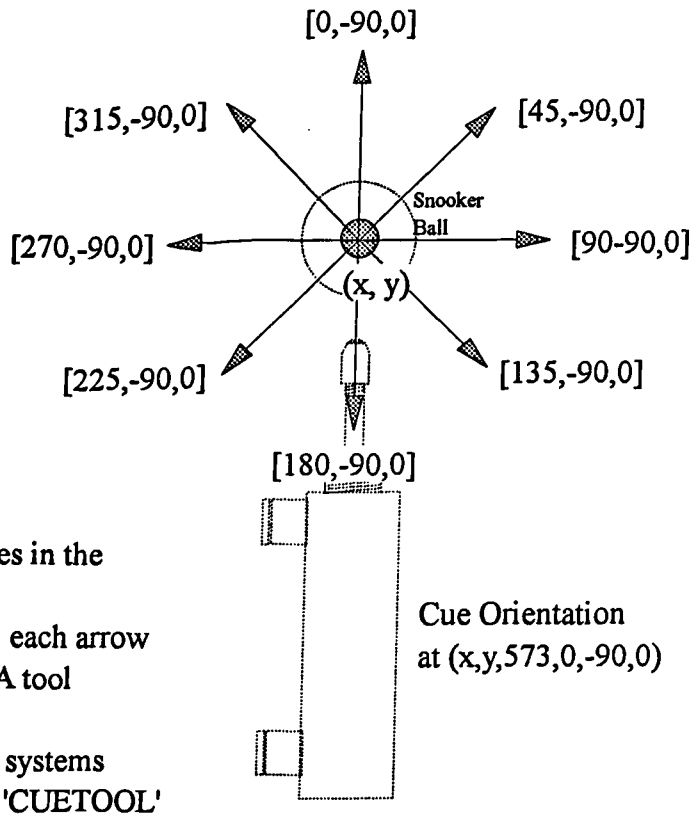
Abduction :

```
IF   ( INRANGE( TRANS(x+20,y,276,-90,-90,0))==0 AND
       INRANGE( TRANS(x+k,y+k,276,-90,-90,0))==0 AND
       INRANGE( TRANS(x,y+20,276,-90,-90,0))==0 AND
       INRANGE( TRANS(x-k,y+k,276,-90,-90,0))==0 AND
       INRANGE( TRANS(x-20,y,276,-90,-90,0))==0 AND
       INRANGE( TRANS(x-k,y-k,276,-90,-90,0))==0 AND
       INRANGE( TRANS(x,y-20,276,-90,-90,0))==0 AND
       INRANGE( TRANS(x+k,y-k,276,-90,-90,0))==0 AND
```

THEN (x,y,276) is within reach of the On-board Camera

Figure A.2 On-board Camera Custom In Range Test





Note : 1. The pneumatic cue always operates in the X-Y plane Z=573

2. In the diagram on the above-right, each arrow represents the X-axis of the PUMA tool 'CUETOOL' co-ordinate system.
The 'CUETOOL' and PUMA base systems X-Y planes are parallel and so the 'CUETOOL' Y and Z axes can be deduced.
The orientation parameters of 'CUETOOL' [O,A,T] are shown in square brackets.

Abduction :

```
IF   ( INRANGE( TRANS(x,y,573,0,-90,0))==0 AND
       INRANGE( TRANS(x,y,573,45,-90,0))==0 AND
       INRANGE( TRANS(x,y,573,90,-90,0))==0 AND
       INRANGE( TRANS(x,y,573,135,-90,0))==0 AND
       INRANGE( TRANS(x,y,573,180,-90,0))==0 AND
       INRANGE( TRANS(x,y,573,225,-90,0))==0 AND
       INRANGE( TRANS(x,y,573,270,-90,0))==0 AND
       INRANGE( TRANS(x,y,573,315,-90,0))==0 AND
```

THEN (x,y,573) is within reach of the Pneumatic Cue

Figure A.3 Pneumatic Cue Custom In Range Test

Appendix B Effect of Deviation from Ideal Line of Pocket Entry on Angular Tolerance

Given a pocket point P_1 and a target ball at T, the line L_1 passing through P_1 and T defines the path of the target ball for a successful shot. However, because the pocket is wider than D, the ball diameter, small deviation from the expected path can be tolerated while the target ball can still be potted, see figure B.1. Point P_2 and P_3 in figure B.1 indirectly define the limit of the tolerable deviation of the target ball path. The line segment P_2P_3 passes through the pocket point P_1 and is normal to L_2, the line bisecting the pocket through P_1. P_2 and P_3 are equi-distance from P_1 and so $|P_1P_2| = |P_1P_3| = k$. Let the angle between L_1 and L_2 equals θ, the deviation of the target ball path from the ideal path. α_1 and α_2 are the angles between P_2T and L_1, and P_3T and L_1 respectively. The total angular tolerance for a successful shot therefore equals $(\alpha_1 + \alpha_2)$. Suppose the distance between T and P_1 equals l, $(\alpha_1 + \alpha_2)$ can be expressed in terms of l and θ.

By first concentrating on the right-angle triangle $P_1P_2P_4$ in figure B.2, knowing that $|P_1P_2| = k$ and the angle between P_1P_2 and P_2P_4 equals θ, $|P_1P_4|$ and $|P_2P_4|$ can be expressed as $k\sin\theta$ and $k\cos\theta$ respectively. And since triangle TP_2P_4 is also a right-angle triangle,

$$\tan \alpha_1 = k \cos\theta / l - k \sin\theta \qquad \textbf{Eq. (B.1)}$$

Similarly, focusing on triangle TP_3P_5

$$\tan \alpha_2 = k \cos\theta / l + k \sin\theta \qquad \textbf{Eq. (B.2)}$$

Thus $(\alpha_1 + \alpha_2)$ can be derived from :

$$\tan(\alpha_1 + \alpha_2) = \frac{\tan \alpha_1 + \tan \alpha_2}{1 - \tan \alpha_1 \tan \alpha_2} \qquad \textbf{Eq. (B.3)}$$

Substituting Eq. (*B.1*) and (*B.2*) into (*B.3*), and after simplification, the following is obtained ;

$$\tan(\alpha_1 + \alpha_2) = \left(2lk/l^2 - k^2\right)\cos\theta$$

$$\Rightarrow \alpha_1 + \alpha_2 = \tan^{-1}\left[\left(2lk/l^2 - k^2\right)\cos\theta\right] \qquad \textbf{Eq. (B.4)}$$

Eq. *(B.4)* shows that the relationship between θ and $(\alpha_1 + \alpha_2)$. A special case of Eq. *(B.4)* occurs when $l = k$, in which case $(l^2 - k^2) = 0$ and so $(\alpha_1 + \alpha_2)$ equals 90° irrespective of θ. Otherwise for any general target ball path, the term $\left(2lk/l^2 - k^2\right)$ can be taken as a non-zero constant. Therefore as the deviation from the ideal path increases, θ increases and the value of $\cos\theta$ decreases. As a result, it can be concluded from Eq. (*B.4*) that the angular tolerance for a successful shot $(\alpha_1 + \alpha_2)$ is inversely proportional to θ, the angle between the ideal line of entry to the pocket and the actual line of entry.

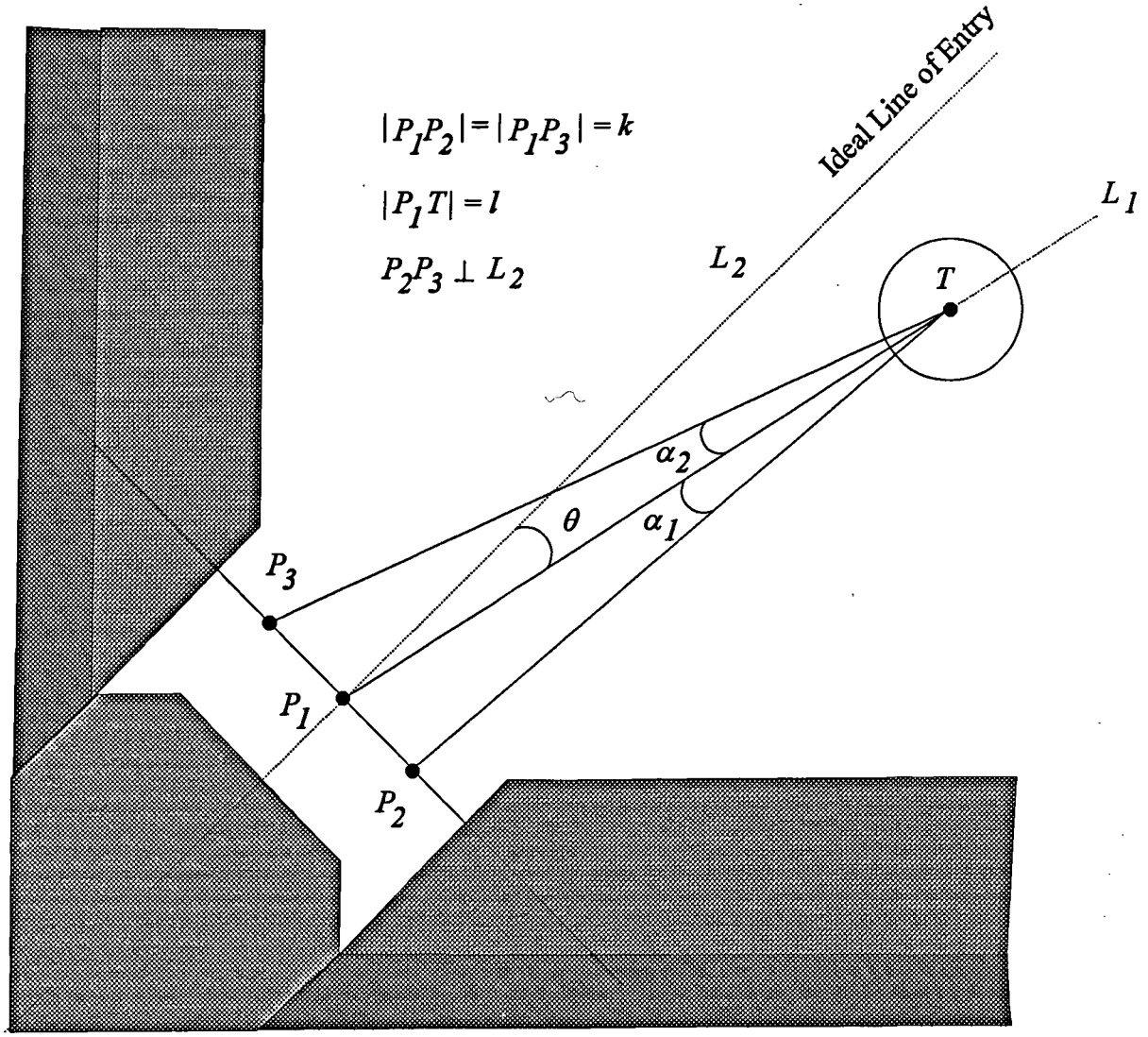


Figure B.1 Angular Tolerance of Target Ball Path

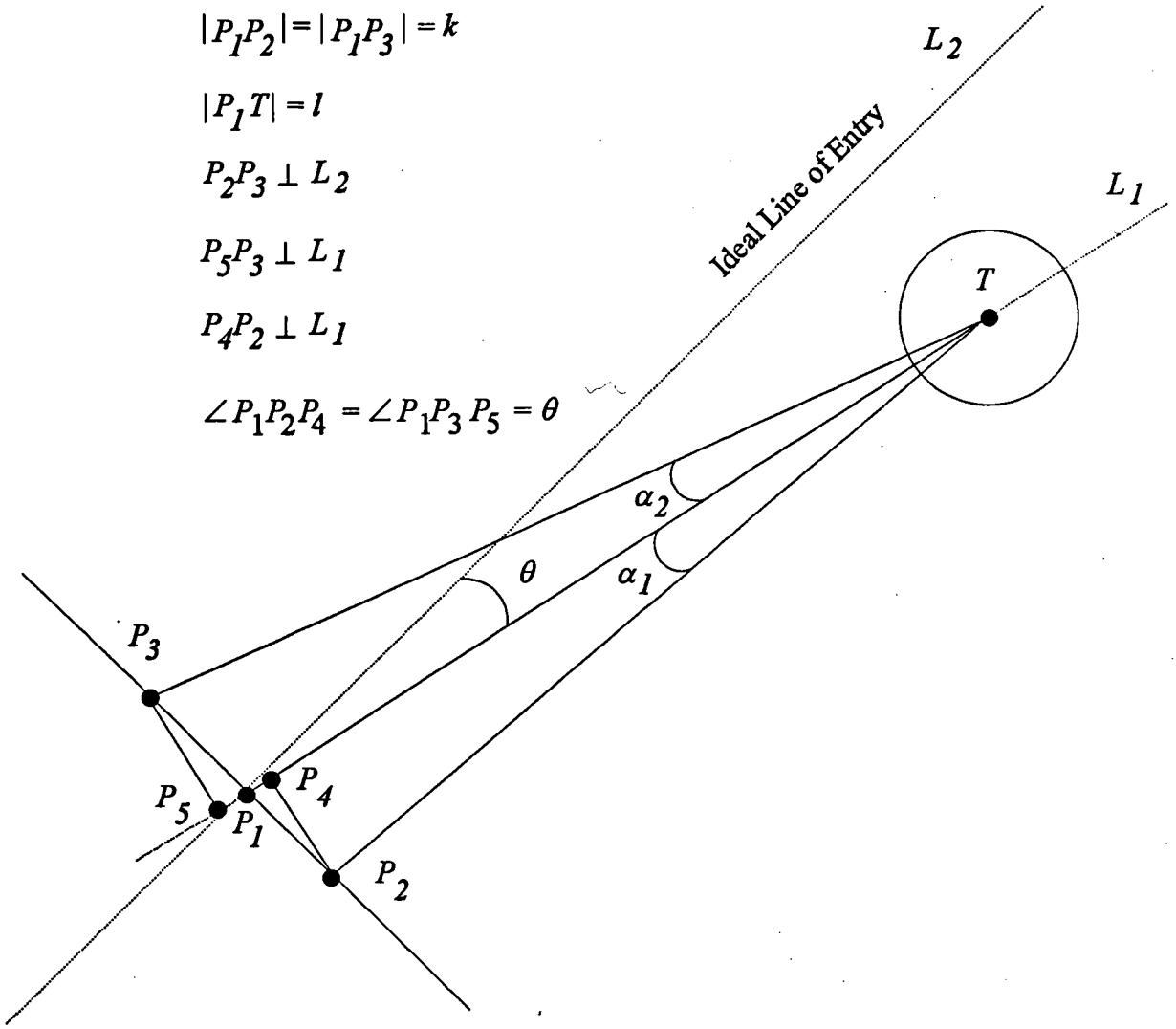


Figure B.2 Expressing Angular Tolerance in Terms of θ and l

Appendix C Conversion Between Pneumatic Cue Force and Initial Ball Velocity

The pneumatic cue force is set according to the value of 8 binary switches. The switches are treated as an unsigned 8-bit word, the value of which ranges from '00000000' (0 decimal) to '11111111' (255 decimal) thus giving 256 discrete settings. For the purpose of ball dynamics modelling and game planning, the pneumatic cue force has to be converted into the equivalent initial cue ball velocity, assuming that the cue strikes the cue ball dead centre horizontally.

The experiment set up to investigate the relationship between the pneumatic cue force setting and the resultant initial cue ball velocity involves the use of high speed video equipment. The basic idea is to determine how long the cue ball takes to traverse a small fixed distance after being hit by the pneumatic cue, thereby inferring its initial velocity.

The SKF linear driver and the PUMA robot arm are both locked in position once the pneumatic cue is lined up horizontally with the cue tip level with the ball centre. A point is marked on the green baize so that the cue ball can be repositioned accurately. Two metal blocks are placed 48 mm apart behind the ball to help identify when the ball has traversed the gap, as shown in figure C.1.

The test procedure is as follows :

(i) Set up the high speed video camera ready for recording.
(ii) Select the required pneumatic cue force (1-256).
(iii) Place the cue ball on the fixed spot.
(iv) Start recording.
(v) Fire the pneumatic cue.
(vi) Stop recording.

Upon play-back of the recorded footage, the elapsed time for the ball to traverse the gap (48 mm) between the marker blocks can be determined from the high speed video clock which is also displayed. The high speed video was set to work at 4000 frames per second. The pneumatic cue was tested from force level 10 up to 250 in steps of 10. The maximum cue force level (256) was also tested. At each force level, the striking of the cue ball was recorded 3 times to eliminate any possible experimental error. The test results are tabulated in table C.1.

Figure C.2 is an X-Y scattered plot of the test results. Using linear regression, the line that best fit the test results is also plotted in the figure. Admittedly, the test results are not truly linear but this is acceptable allowing for factors such as minor variations in the supply of compressed air and the repeatability of the pneumatic cue. It can therefore be concluded that the initial cue ball velocity increases linearly with the pneumatic cue force level.

Having established the linear relationship between the pneumatic cue force level and the initial ball velocity, a mapping can be drawn up as in figure C.3. Using this Force-Velocity map, the initial ball velocity corresponding to a given pneumatic cue force level can be computed as :

$$V(n) = V_{\max} \times {}^{n}/_{256} \qquad \text{where } n = \text{the selected cue force level,}$$

$1 \le n \le 256$, n $\in$ integer

$V_{\max}$ = cue ball velocity at maximum cue force

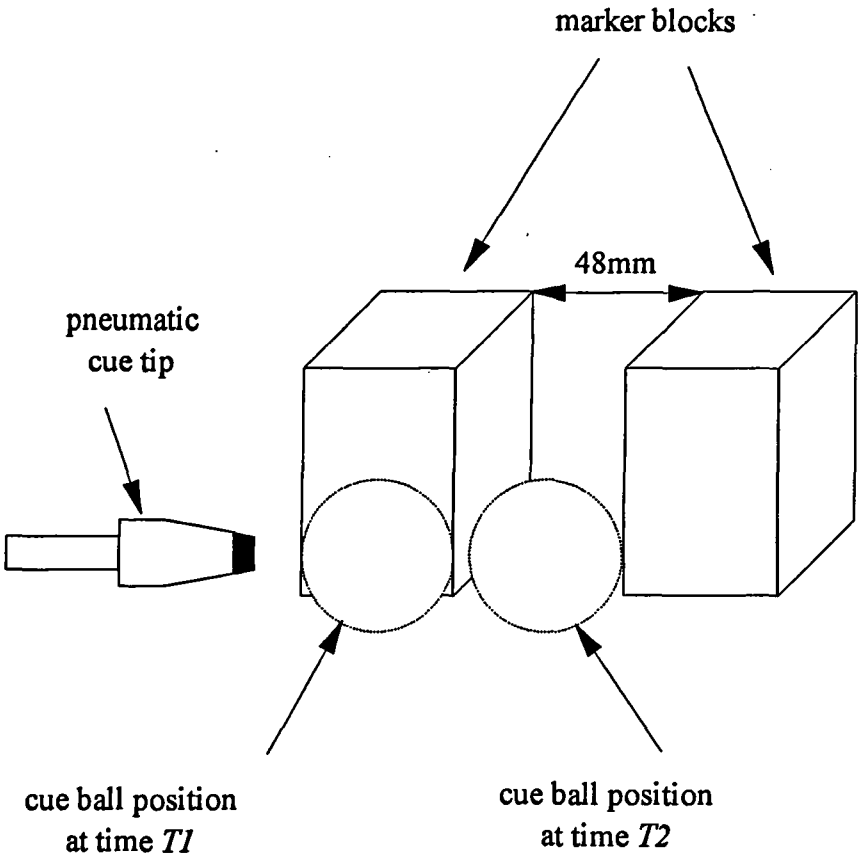
Given an initial cue ball velocity, the corresponding required pneumatic cue force can be computed as follows :

$$F(v) = trunc(256 \times {}^{v}/_{V_{\max}} + 0.5)$$

where v = initial ball velocity

$0 < v \le V_{\max}$, $v \in$ real

The expression inside the brackets above has to be truncated because function F must return an integer. The function '*trunc*' removes the decimal part of a real number, hence 0.5 is added to round the real number to the nearest integer.



Time taken to travelled 48mm = *T2* - *T1*

Figure C.1 Experimental Set-up to Determine Initial Cue Ball Velocity

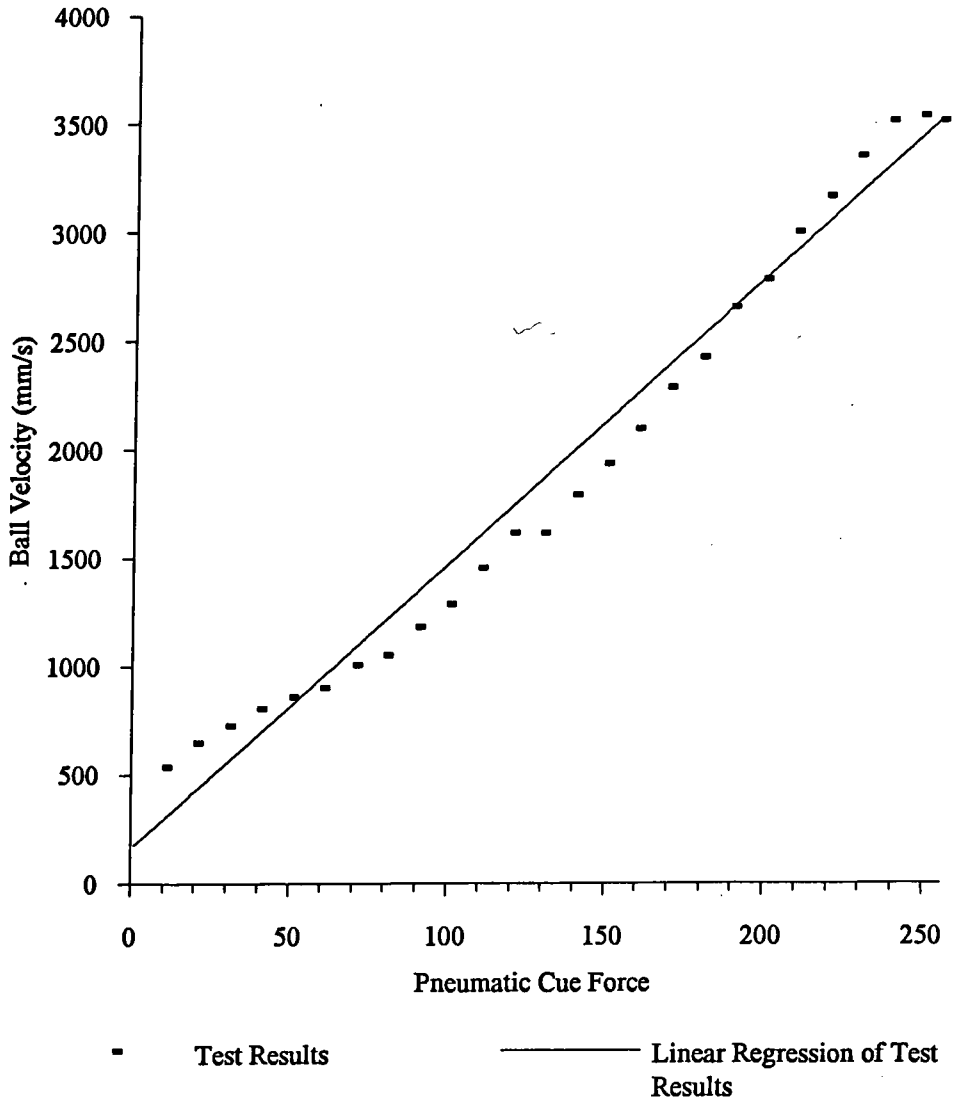


Figure C.2 Initial Ball Velocity-vs-Pneumatic Cue Force

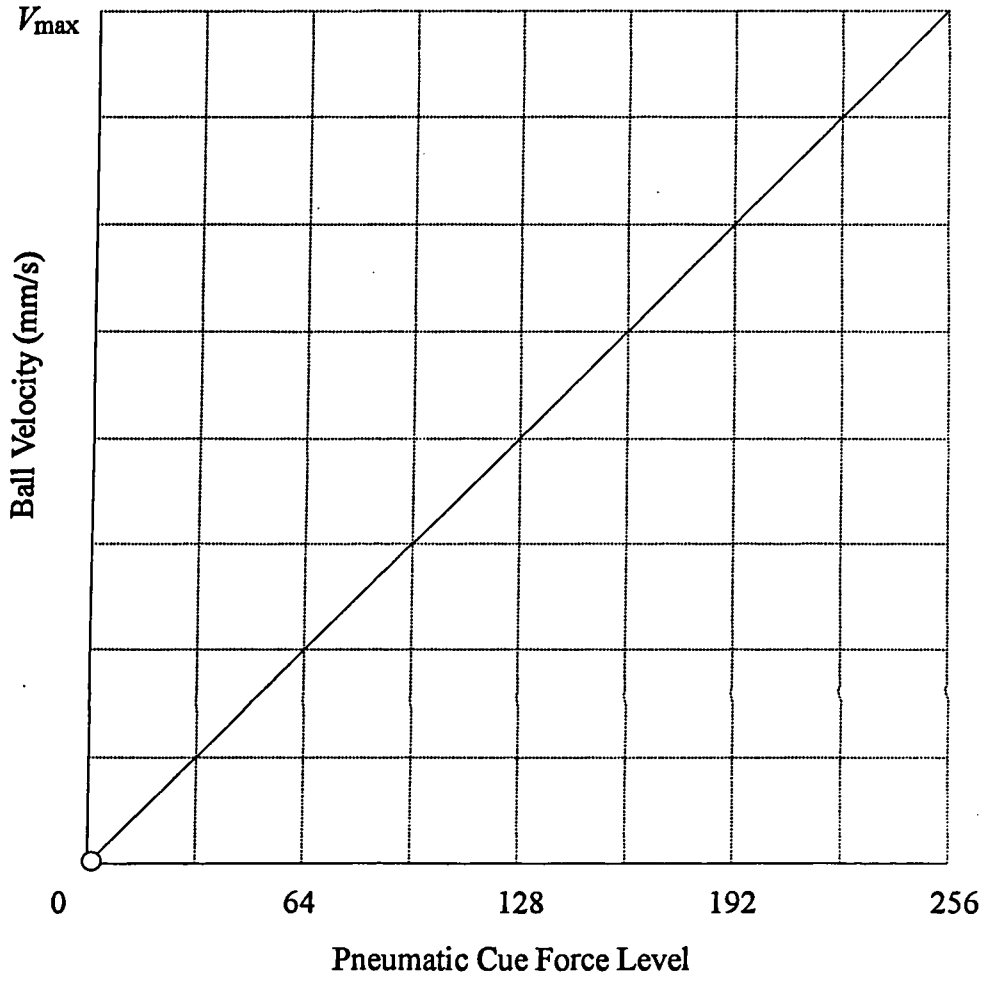


Figure C.3 Linear Modelling of the Relationship Between Pneumatic Cue Force and Initial Ball Velocity

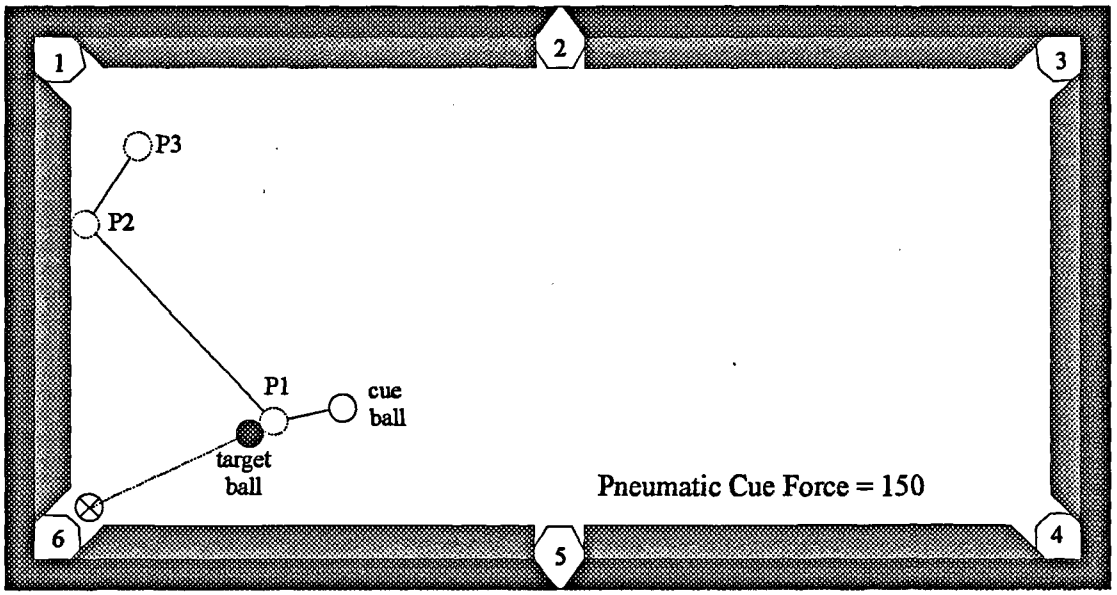
Pneumatic Cue Force Level	Mean time taken to traverse 48 mm (seconds)	Ball Velocity (mm/s)
10	0.08970	535.12
20	0.07430	646.03
30	0.06600	727.27
40	0.05970	804.02
50	0.05600	857.14
60	0.05330	900.56
70	0.04770	1006.29
80	0.04570	1050.33
90	0.04070	1179.36
100	0.03730	1286.86
110	0.03300	1454.55
120	0.02970	1616.16
130	0.02970	1616.16
140	0.02680	1791.04
150	0.02480	1935.48
160	0.02290	2096.07
170	0.02100	2285.71
180	0.01980	2424.24
190	0.01810	2651.93
200	0.01725	2782.61
210	0.01600	3000.00
220	0.01517	3164.14
230	0.01433	3349.62
240	0.01367	3511.34
250	0.01358	3534.61
256	0.01367	3511.34

Table C.1 Experimental Results on Initial Ball Velocity Trials

Appendix D Example Shots Used in Fitness Assessment

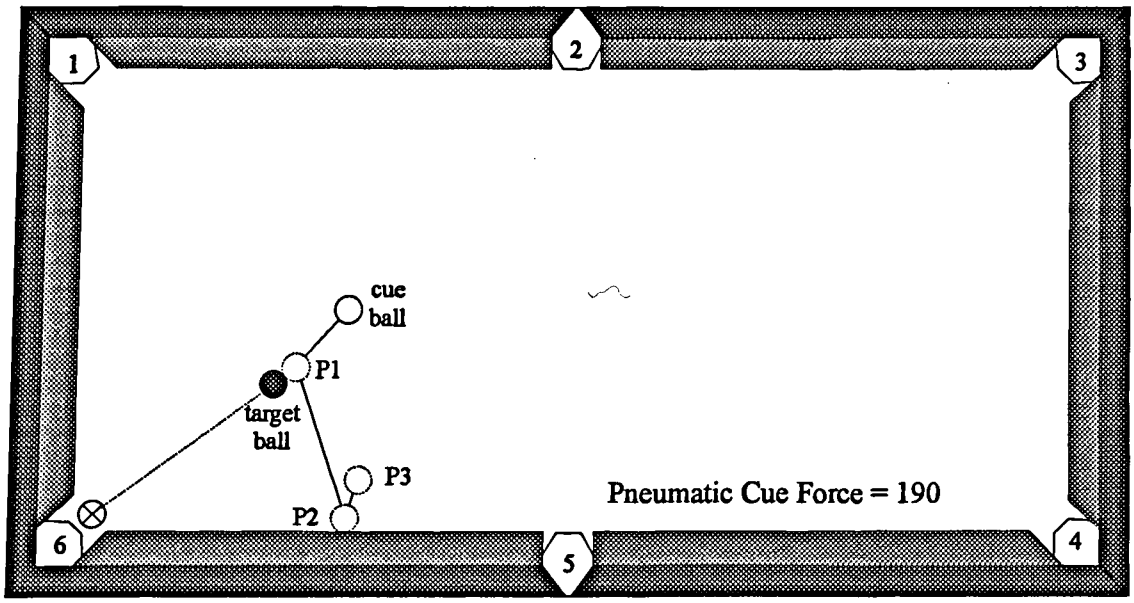
Example shots are recordings of what actually happens when the Robot Snooker Player plays a shot using a certain pneumatic cue force. They are required so that the correctness of the cue ball path prediction made by the dynamics model can be assessed. The 5 example shots are listed below.

Example Shot No.1



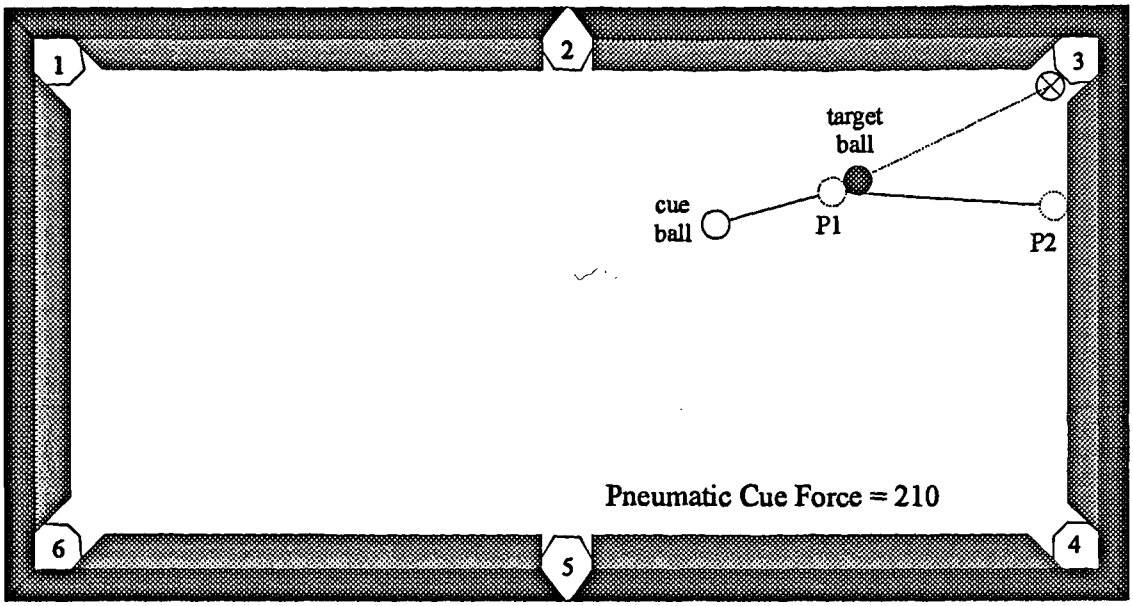
	World Coordinates (x, y)
Cue Ball	(453.30, 566.76)
Target Ball	(307.21, 601.90)
Pocket No.6	(51.34, 726.92)
P1	(346.93, 582.49)
P2	(50.17, 271.70)
P3	(121.09, 138.88)

Example Shot No.2



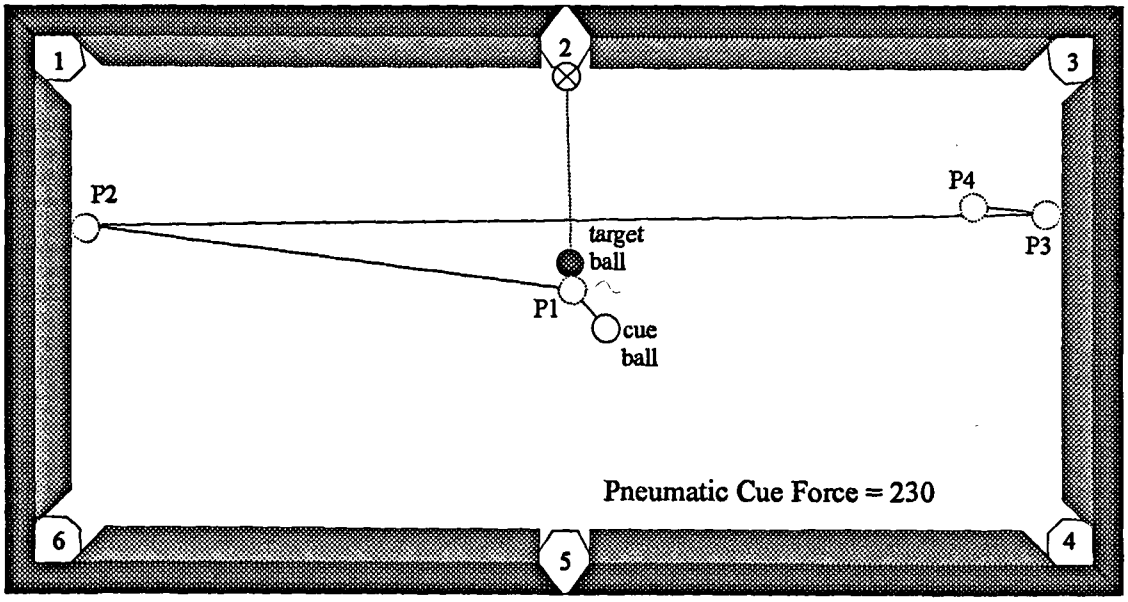
	World Coordinates (x, y)
Cue Ball	(446.68, 397.46)
Target Ball	(332.24, 509.04)
Pocket No.6	(51.34, 726.92)
P1	(367.18, 481.94)
P2	(442.05, 728.99)
P3	(465.28, 674.98)

Example Shot No.3



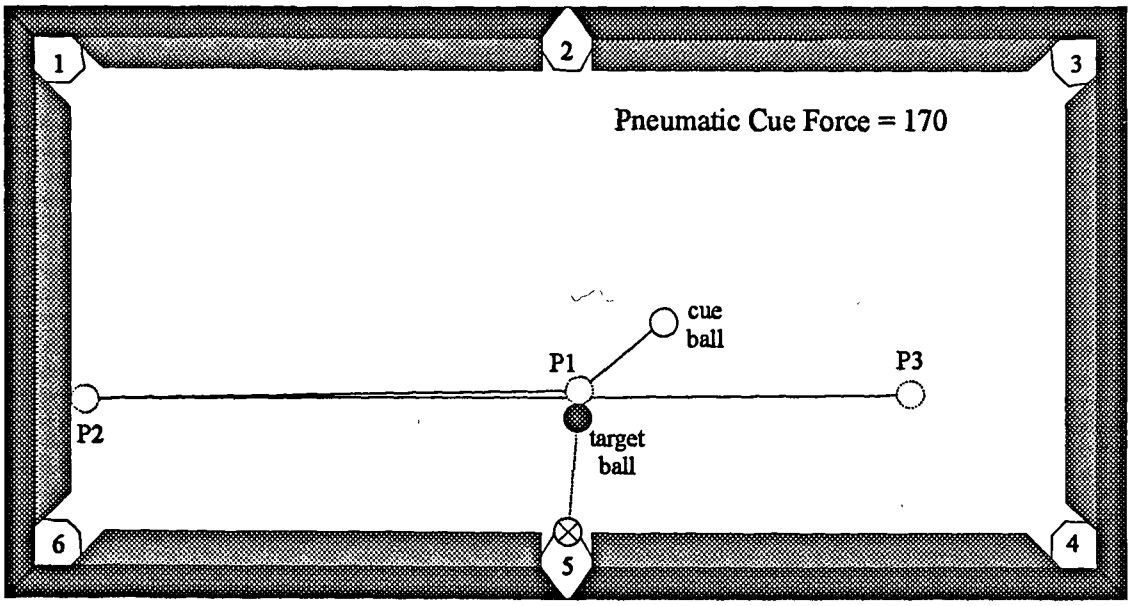
	World Coordinates (x, y)
Cue Ball	(1023.97, 261.77)
Target Ball	(1253.63, 191.45)
Pocket No.3	(1545.78, 56.09)
P1	(1213.51, 210.04)
P2	(1548.33, 432.96)

Example Shot No.4



	World Coordinates (x, y)
Cue Ball	(858.81, 436.11)
Target Ball	(801.60, 324.18)
Pocket No.2	(799.28, 29.15)
P1	(801.95, 368.38)
P2	(49.69, 284.55)
P3	(1550.00, 239.34)
P4	(1452.78, 237.26)

Example Shot No.5



	World Coordinates (x, y)
Cue Ball	(952.09, 418.09)
Target Ball	(814.66, 570.30)
Pocket No.5	(802.50, 740.31)
P1	(817.81, 526.20)
P2	(49.48, 543.01)
P3	(1331.00, 523.07)

References

[Allen91] Peter K. Allen, Billibon Yoshimi, and Aleksandar Timcenko, "Real-Time Visual Servoing", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp.851-856, April 91.

[Andersson89] Russell L. Andersson, "Dynamic Sensing in a Ping-Pong Playing Robot", IEEE Transactions on Robotics and Automation, vol.5, no.6, pp.728-739, December 89.

[Barricelli62] Barricelli, N.A. (1962) Numerical Testing of Evolution Theories, ACTA Biotheoretica 16 p.69-126

[Birk81] John R. Birk and Robert B. Kelley, "An Overview of the Basic Research Needed to Advance the State of Knowledge in Robotics", IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-11, no.8, pp.574-579, August 81.

[Bowman87] M.E. Bowman and A.K. Forrest, "Transformation Calibration of a Camera Mounted on a Robot", Image and Vision Computing, vol.5, no.4, pp.261-266, November 1987.

[Bowman88] M.E. Bowman and A.K. Forrest, "Visual Detection of Differential Movement : Applications to Robotics", Robotica (1988) vol.6, pp.7-12.

[Chen87] Jigien Chen and Lih-Ming Chao, "Positioning Error Analysis for Robot Manipulators with All Rotary Joints", IEEE Journal of Robotics and Automation, vol.RA-3,no.6, pp.539-545, December 1987.

[Cheung90] W. Cheung, K. Khodabandehloo, and I.J. Rennell, "Development of Expert Robot Systems for Skilled Operation", Proceedings of the 28th International MATADOR Conference 1990, pp.41-47.

[Crowley85] James L. Crowley, "Navigation for an Intelligent Mobile Robot", IEEE Journal of Robotics and Automation, vol. RA-1, no.1, pp.31-41, March 85.

[Davis 88] Steve Davis et al, Matchroom Snooker, Pelham Books 1988.

[Davis90] Lawrence Davis, "Hybrid Genetic Algorithms for Machine Learning", IEE Colloquium on "Machine Learning", Computing and Control Division, 28 June 1990.

[Frost90] A.R. Frost, "Robotic Milking : a review", Robotica (1990) vol.8, pp.311-318.

[Goldberg89] Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimisation, and Machine Learning, Addison Wesley Publishing Company, Inc.

[Gonzalez87] Gonzalez, R.C. and Wintz, P. (1987) Digital Image Processing, Addison Wesley Publishing Company, Inc, pp.177.

[Holland75] Holland J. (1975) Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.

[Hashimoto91] Koichi Hashimoto, Tsutomu Kimoto, Takumi Ebine, and Hidenori Kimura, "Manipulator Control with Image-Based Visual Servo", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp.2267-2272, April 91.

[Jang91 (I)] Won Jang and Zeungnam Bien, "Feature-based Visual Servoing of an Eye-In-Hand Robot with Improved Tracking Performance", Proceedings of the 1991 IEEE International Conference on Rcbotics and Automation, pp.2254-2260, April 91.

[Jang91 (II)] Won Jang, Kyungjin Kim, Myungjin Chung, and Zeungnam Bien, "Concepts of augmented image space and transformed feature space for efficient visual servoing of an 'eye-in-hand robot'", Robotica (1991) vol 9 pp 203-212.

[Judd90] Robert P. Judd and Al B. Knasinski, "A Technique to Calibrate Industrial Robots with Experimental Verification", IEEE Transactions on Robotics and Automation, vol.6, no.1, pp.20-30, February 1990.

[Kassler90] Michael Kassler, "Introduction to the special issue on robots and food-handling", Robotica (1990) vol. 8, pp.267-268.

[Khalil91] W. Khalil, M. Gautier, and Ch. Enguehard, "Identifiable parameters and optimum configurations for robots calibration", Robotica (1991) vol 9 pp 63-70.

[Khodabandehloo87] K. Khodabandehloo, I.J. Rennell, and K.H.L. Ho, "Robots with Artificial Vision and Intelligence", Advanced Robotics Programme (ARASA) International Conference on Nuclear Robotic Technologies and Applications Present and Future, 29 June - 1 July 87.

[Khodabandehloo90] K. Khodabandehloo, "Robotic handling and packaging of poultry products", Robotica (1990) vol. 8, pp.285-297.

[Khodabandehloo91] K. Khodabandehloo, "Skilled Robotics", Proceedings of the Second International Conference on Automation, Robotics, and Computer Vision 1992, pp.RO-15.1.1-RO-15.1.5.

[Kim89] D.H. Kim, K.H. Cook, and J.H. Oh, "Identification and compensation of robot kinematic parameter for positioning accuracy improvement", Robotica (1991) vol 9 pp 99-105.

[King88] F.G. King, G.V. Puskorius, F. Yuan, R.C. Meier, V. Jeyabalan, and L.A. Feldkamp, "Vision Guided Robots for Automated Assembly", Proceedings of the 1988 International Conference on Robotics and Automation vol.3, pp.1611-1616.

[Lenz88] Reimar K. Lenz and Roger Y. Tsai, "Techniques for Calibration of the Scale Factor and Image Centre for High Accuracy 3-D Machine Vision Metrology", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.10, no.5, pp.713-720, September 1988.

[Luk91] B.L. Luk, A.A. Collie, J. Billingsley, "ROBUG II : An Intelligent Wall Climbing Robot", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp.2342-2347, April 91.

[Nitzan85] David Nitzan, "Development of Intelligent Robots : Achievements and Issues", IEEE Journal of Robotics and Automation, vol. RA-1, no.5, pp.3-13, March 85.

[Penna91] Michael A. Penna, "Camera Calibration : A Quick and Easy Way to Determine the Scale Factor", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.13, no.12, pp.1240-1245, December 1991.

[Purnell90] G. Purnell, N.A. Maddock and K. Khodabandehloo, "Robot deboning for beef forequarters", Robotica (1990) col. 8, pp.303-310.

[Renders91] Jean-Michel Renders, Eric Rossignol, Marc Becquet, and Raymond Hanus, "Kinematic Calibration and Geometrical Parameter Identification for Robots", IEEE Transactions on Robotics and Automation, vol.7, no.6, pp.721-731, December 1991.

[Rosenberg67] Rosenberg, R.S. (1967) Simulation of Genetic Populations with Biochemical Properties, Dissertation Abstracts Internation 27(7), 2732B.

[Shiu89] Yiu Cheung Shiu and Shaheen Ahmad, "Calibration of Wrist-Mounted Robotic Sensors by Solving Homogenious Transform Equations of the Form AX=XB", IEEE Transactions on Robotics and Automation, vol.5, no.1, pp.16-29, February 1989.

[Su91], Chau Su and Yuan F. Zheng,"Task Decomposition for a Multilimbed Robot to Work in Reachable but Unorientable Space", IEEE Transactions on Robotics and Automation, vol.7, no.6, pp.759-770, December 1991.

[Trevelyan89] James P. Trevelyan, "Sensing and Control for Sheep-Shearing Robots", IEEE Transactions on Robotics and Automation, vol.5, no.6, pp.716-727, December 89.

[Tsai87] Roger Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", IEEE Journal of Robotics and Automation, vol. RA-3, no.4, pp.323-344, August 1987.

[Tsai88] Roger Y. Tsai and Reimar K. Lenz, "Real Time Versatile Robotic Hand/Eye Calibration using 3D Machine Vision", Proceedings 1988 IEEE International Conference on Robotics and Automation vol.1 pp.554-561.

[Tsai89] Roger Y. Tsai and Reimar K. Lenz, "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration", IEEE Transactions on Robotics and Automation, vol.5, no.3, pp.345-358, June 1989.

[Veitschegger88] William K. Veitschegger and Chi-Haur Wu, "Robot Calibration and Compensation", IEEE Journal of Robotics and Automation, vol.4, no.6, pp.643-656, December 1988.

[Weiss87] Lee E. Weiss, Arthur C. Sanderson, and Charles P. Neuman, "Dynamic Sensor-Based Control of Robots with Visual Feedback", IEEE Journal of Robotics and Automation, vol. RA-3, no.5, pp.404-545, October 1987.

[Westmore91] David B. Westmore and William J. Wilson, "Direct Dynamic Control of a Robot Using an End-Point Mounted Camera and Kalman Filter Position Estimation", Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp.2376-2384, April 91.

[Wijesoma93] S.W. Wijesoma, D.F.H. Wolfe, and R.J. Richards, "Eye-to-Hand Coordination for Vision-Guided Robot Control Applications", The International Journal of Robotics Research, vol.12, no.1, pp.65-78, Feb 1993.

[Zheng91] Jiang Yu Zheng, Qian Chen, and Saburo Tsuji, "Active Camera Guided Manipulation", Proceedings of the 1991 IEEE International Conference on Robotics and Automation pp.632-638, April 91.

[Zhuang92] Hanqi Zhuang and Zvi S. Roth, "Robot Calibration using the CPC Error Model", Robotics and Computer-Integrated Manufacturing, vol.9, no.3, pp.227-237, 1992.